HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

BACHELORARBEIT

Frau
Sophie Donhauser

Konzeption und Umsetzung eines modularen Systems zur Generierung virtueller Umgebungen

2014

Fakultät: Medien

BACHELORARBEIT

Konzeption und Umsetzung eines modularen Systems zur Generierung virtueller Umgebungen

Autorin:
Frau Sophie Donhauser

Studiengang:
Angewandte Medien

Seminargruppe:
AM11wD1-B

Erstprüfer:
Prof. Alexander Marbach

Zweitprüfer:
Tobias Tauscher

Faculty of Media

BACHELOR THESIS

Conceptual design and implementation of a modular system for virtual enviroments

author:
Ms. Sophie Donhauser

course of studies:
Applied Media

seminar group:
AM11wD1-B

first examiner:
Prof. Alexander Marbach

second examiner:
Tobias Tauscher

submission:
Mittweida, 08.07.2014

Bibliografische Angaben

Donhauser, Sophie:

Konzeption und Umsetzung eines modularen Systems zur Generierung virtueller Umgebungen

Conceptual design and implementation of a modular system for virtual enviroments

74 Seiten, Hochschule Mittweida, University of Applied Sciences,
Fakultät Medien, Bachelorarbeit, 2014

Abstract

In der vorliegenden Arbeit wird das Thema Konzeption und Implementierung eines modularen Systems in virtuelle Umgebungen untersucht. Sie greift hierfür auf die Rapidbox-Bauweise der Forschungsgruppe Gamecast aus Mittweida zurück. Die Aufgaben der Arbeit sind: Die Konzeption eines Workflows zur Erstellung eines modularen Assetkits, die Entwicklung eines Konzeptes für die Erweiterung des Assetkits, die Umsetzung und Implementierung des Assetkits mit dem konzipierten System RapidSeason in die Shark-Engine und das definieren weiterer Ansätze für Systementwicklungen innerhalb des Rapidbox-Systems. In der Vorbereitung zur Kon-

zeption wird sich auf gängige Workflows und Ansätze von Paul Mader. Lee Perry, Tyler Wanlass und Joel Burgess berufen. Techniken von Tor Frick sowie Joshua Kinney werden ebenfalls hinzugezogen, um ein möglichst breites Spektrum an Arbeitsweisen zu garantieren. Der herauskristallisierte Workflow wurde mit Hilfe der Shark3D-Engine durch das Thema „Haus im Wald“ getestet. Zusätzlich wurde das erweiterte System RapidSeason, was die Assets um Jahreszeiten bereichert, integriert. Dadurch konnte der repetitive Charakter von modularen Assetkits aufgebrochen werden und eine Wiederverwertbarkeit sowie Lebendigkeit geschaffen werden. An diesem Punkt ist es denkbar, dass weitere Systeme wie RapidSeason entwickelt werden. Ansätze dafür sind in dieser Arbeit ebenfalls enthalten. Diese Bachelorarbeit setzt Wissen im Bereich 3D-Design für Games voraus, sowie das Verständnis für englischsprachige Texte.

Inhaltsverzeichnis

Abkürzungsverzeichnis

Anm. d. Verf.	Anmerkung der Verfasserin
Shark3D	Shark-3D-Engine
Lowpoly/LowPoly	Low-Poly-Geometrie
Highpoly/HighPoly	High-Poly-Geometrie

Glossar

Asset	Ein Teil, eines modularen Baukastens.
Assetkit	Ein modularer Baukasten, der auf einem vordefinierten Raster-ausgerichtet ist. Danach unterscheidet sich auch die Skalierung seiner Teile.
Props	Ein Teil, welches nicht auf dem Raster ausgerichtet ist und frei platziert wird.
Subkits	Ein modularer Baukasten, der in sich geschlossen ist. Zum Beispiel ein Dachkit mit End- und Mittelstücken.
Vertex	Ein Scheitelpunkt, der aus den drei Koordinaten X,Y und Z besteht. Ein Vertex dient zur Modifizierung eines 3D-Objektes.
Polygon	Eine Fläche aus drei oder mehr Vertices. Dreieckige, Tris, und Viereckige, Quads werden zur Modellierung genutzt.
GTA V	Ein Computerspiel von der Firma Rockstar, welches insbesondere auf eine offene Spielwelt setzt.
I am Alive	Ein Computerspiel von Ubisoft. Die Szenerie ist eine postapokalyptische Welt.
Skyrim	Ein Computerspiel von Bethesda, welches eine offene Spielwelt besitzt und modular aufgebaut ist.
Gamecast	Eine Forschungsgruppe in Mittweida, welche sich mit der Entwicklung von Prävisualisierungssoftware beschäftigt.
Cinector	Eine Software, die von der Forschungsgruppe Gamecast entwickelt wurde und zur Prävisualisierung eingesetzt wird.
Engine	Eine Software, die als Levelgenerator fungiert
Modding	Das Erstellen von veränderten oder neugenerierten Spielinhalten für ein veröffentlichtes Computerspiel.

RapidBox	Eine modulare Bauweise, die bei der Forschungsgruppe Gamecast entwickelt wurde.
RapidSeasons	Ein Jahreszeitensystem, was auf Assetzuständen fußt.
Kitbashing	Ein Verfahren von Bethesda, welches es ermöglicht Assetkits miteinander zu kombinieren und Crossover-Bereiche zu schaffen.
Graybox-Phase/ Blocking Out	Eine Produktionsphase in der Assetkiterstellung, bei der durch primitive Geometrie einen ersten Leveleindruck vermittelt.
Glue-Kit	Modulare Baukästen, die speziell für Übergänge konzipiert wurden. Selbiges bezeichnet ebenfalls die Transitional Pieces von Perry.
Character	Vom Spieler gesteuerte virtuelle Figur.
Maps	Verschiedene Texturen, die durch ihre Farbgebung durch ihre Anwendung auf einem 3D-Objekt seine Eigenschaften näher definieren. So definiert eine Specularmap z.B. die Positionierung und Stärke der Spitzlichter.
Mesh	Die geometrische Form eines 3D-Objektes.
GUI/ GUI elements	Graphical User Inferface (elements), d.h. die Benutzeroberfläche eines Programms.
Loopen	Das Aneinandersetzen gleicher Teile, z.T. sogar bis zur Unendlichkeit, auch oft Cycling genannt.
Tonality/Tonalität	Die Stimmung, die im Marketing mit ausgewählten Attributen für eine Werbekampagne oder aber einzelne Bilder deifniert wird.

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Problemstellung

In der Gamesbranche und der virtuellen Filmproduktion existiert ein sehr hoher Zeit- bzw. Kostendruck. An Spielen wie z.B. Skyrim oder GTA 5 werden zum Teil Jahre mit mehrköpfigen Teams entwickelt. Hierbei muss das Budget nicht nur den Produktionsprozess sondern auch die Marketing- sowie Vertriebskosten abdecken. Gerade in diesen Open-World-Games entsteht ein hoher Aufwand, da viele Assets produziert werden müssen, um die Größe der Spielwelt zu ermöglichen. Wie auch in der realen Welt muss die virtuelle Welt erscheinen, als sei kein Blatt wie das andere. Nur so kann sie für den Spieler durch ihre Detailtiefe bestechen und glaubhaft wirken.

Desweiteren sind finanzielle Mittel nicht unbegrenzt und schränken somit den Produktionszeitraum ein. Durch diese begrenzte Zeitspanne wird insbesondere bei den 3D-Grafiken nach Optimierungen gesucht. Eine modulare Bauweise ist hierbei oft die Lösung für die notwendigen Assets, kombiniert mit individuellen Einzelelementen, den Props. Innerhalb der Produktion werden sogenannte Assetkits erstellt, die immer wieder miteinander kombiniert werden können. So entsteht ein Vielfaches an Kombinationsmöglichkeiten, welche Kosten und Zeit sparen, dem Spieler aber trotzdem eine abwechslungsreiche Spielwelt bieten.

Assetkits setzen sich also - wie ein Lego-Baukasten - aus verschiedenen Einzel- bzw. Grundteilen zusammen und ergänzen einander zu neuen Objekten. So wird aus einer Wand, Decke, Fenster und Dach ein Haus. Gibt es nun noch mehrere Varianten von z.B. Wänden oder Fenstern wird es somit möglich verschiedene Arten von Häusern zu bauen. Trotz dieser Kombinationsmöglichkeiten kommt es vor, dass modulare Bausteine dem Spieler auffallen und das Level so repetitiv wirken lassen können. Auf Basis dessen analysierte Scott Jones in seiner Bachelorarbeit wie Modularität in existierenden Spielen "versteckt" wird[1].

Die modulare Bauweise, die für Assetkits verwendet wird, unterliegt immer vordefinierten Regeln, wie Raster, Maßeinheit und der Form der Einzelelemente – es stellt sich die Frage wie die Wiederverwendbarkeit der Asset bzw. Assetkits weiter erhöht werden kann. Der Nachteil, wenig individuelle Levels durch die modulare Bauweise vorzufin-

[1] Vgl. Jones, 2011

den und den Spieler somit visuell Eintönigkeit zu präsentieren, liegt bei sich stets wiederholenden Levelbausteinen nahe.

Dieser Problemstellung widmet sich diese Bachelorarbeit.

1.2 Zielsetzung

Wie bereits in Kapitel 1.1 erwähnt, droht Spielleveln visuelle Eintönigkeit bei der Verwendung von modularen Bauweisen. Daraus resultiert die Zielsetzung die Wiederverwendbarkeit von Assetkits zu erhöhen und dennoch eine Erhöhung der Detailtiefe zu gewährleisten.

Hierzu soll im Rahmen dieser Arbeit ein Ansatz erarbeitet werden, der durch die Einhaltung des resultierenden Konzeptes die Lebendigkeit des modularen Assetkits für die Erstellung eines virtuellen Filmsets erhöht. Hierfür werden Jahreszeiten als Themenvorgabe gewählt und das Assetkit so visuell variiert.

Das Potential auf diese Weise nicht nur die Wiederverwendbarkeit der Assets sondern auch ihre ästhetische Wirkung zu erhöhen ermöglicht es weitere Kosten in der Produktion zu sparen. Zudem erhält dieser modulare „Lego-Baukasten“ auch durch Vegetation und Witterungseinflüsse Natürlichkeit, die sonst nicht gegeben wäre.

Abschließend wird das Assetkit auf der Grundidee von Rapidbox, einer modularen Bauweise von der Forschungsgruppe Gamecast, angesiedelt. Das Resultat der Praxisleistung sollt hierbei in die Engine der Forschungsgruppe eingebunden werden, sodass es als virtuelles Filmset genutzt werden kann.

1.3 Motivation

Das Berufsfeld des 3D-Designers zeichnet sich einerseits durch gestalterische sowie technische Kompetenz aus. Innerhalb der Computerspielbranche werden hierbei verschiedene Spezialisierungen in Form von Character Design und Environment Design & Props von Großunternehmen dem Generalisten vorgezogen.

Die Faszination mit Hilfe von dreidimensionalen Formen eine lebendig wirkende Welt zu erschaffen, die Spieler fasziniert und in ihren Bann zieht, war hierbei die Motivation, mich dem Environment Design zuzuwenden. Das zugrunde liegende Thema der Bachelorarbeit bildet demzufolge für mich eine neue Möglichkeit, Modularität als elementaren Bestandteil des Workflows in ein eigenes Projekt zu integrieren.

Im Laufe meines Praktikums bei Gamecast erlangte ich diesbezüglich bereits erstes technisches sowie gestalterisches Know How. Die Bachelorthesis dient hierbei dazu, dieses Wissen weiter zu vertiefen und das Rapidbox-Konzept der Forschungsgruppe weiterzuentwickeln. Somit entsteht auch für andere 3D-Artists ein Mehrwert, der die Asset Creation für sie optimieren bzw. vereinfachen kann.

1.4 Gliederung und Vorgehensweise

Zuerst wird innerhalb des ersten Teils der Arbeit durch die Zugrundelegung der Theorie das Wissen erarbeitet, welches für die Erarbeitung des weiterentwickelten Konzeptes notwendig ist. Dies erfolgt durch eine Literaturrecherche und zieht Onlinequellen hinzu. Nach der Analyse bestehender Ansätze und erfolgter Konzeptausarbeitung wird das Assetkit erstellt und anhand eines praktischen Beispiels, in diesem Fall einem Gebäude, getestet. Abschließend wird das Assetkit um die vier Jahreszeiten erweitert und somit eine Möglichkeit zur Wiederverwendbarkeit geschaffen.

Für einen besseren Überblick hierbei folgt eine Aufschlüsselung der einzelnen Kapitel:

Im ersten Kapitel wird die Problemstellung erläutert, welche 3D-Design und die damit verbundene "Asset-Creation" mit sich bringt. Durch die Zielsetzung wird definiert, inwiefern in dieser Arbeit ein Lösungsansatz ausgearbeitet werden kann. Hinzu kommen der persönliche Aspekt sowie eine Zusammenfassung für die weitere Vorgehensweise in der Arbeit.

Im zweiten Kapitel wird das Grundwissen vermittelt, welches für das tiefere Verständnis der Arbeit benötigt wird. Zu Beginn wird Modularität in ihrer Definition beschrieben. Es werden Bezüge zu Bereichen herausgearbeitet, welche modulare Elemente innerhalb ihrer Produktion verwenden und wie diese eingesetzt werden. Ein Beispiel hierfür sind die vorgefertigten Häuser in der Architektur. Des Weiteren wird nun näher auf Leveldesign für Games und somit auch virtuelle Umgebungen eingegangen und der Bezug zu dem genutzten System erklärt. Der geschichtliche Rahmen mit Ansätzen von Paul Mader, Lee Perry, Tyler Wanlass mit Ergänzungen von Joshua Kinney sollen ebenfalls Erwähnung finden.

Im dritten Kapitel wird festgelegt, welche technischen Begrenzungen vorliegen und inwiefern die Tools Anforderungen an das Asset stellen. Es sollen hierbei auch die besonderen Merkmale der Shark-3D-Engine herausgearbeitet und die Ergänzungen von Cinector aufgeschlüsselt werden.

Im vierten Kapitel werden die Konzeption und die Umsetzung des Praxisprojektes beschrieben. Zudem findet eine Erläuterung des Arbeitsablaufes statt.

Im fünften Kapitel soll das System "RapidSeasons" beschrieben werden. Die Kernaspekte bilden die vier Jahreszeiten mit ihren spezifischen Eigenschaften in Bezug auf das Erscheinungsbild des Gebäudes sowie der Vegetation. Abschließend erfolgt die Darstellung des Praxisprojektes innerhalb der Engine.

Im sechsten Kapitel erfolgt ein Ausblick über den zukünftigen Einsatz des Systems und der zugrunde liegenden Logik für Assets in Cinector. Des Weiteren wird eine Erweiterung des Systems vorgeschlagen.

Im siebten Kapitel ergeben sich die Zusammenfassung der Arbeit sowie das Resümee. Es wird auf weitere Möglichkeiten zur Forschung hingewiesen und Erkenntnisse zusammengetragen.

2 Hintergrundwissen und Definition

2.1 Bezüge und Definition der Modularität

Modularität ist keine Idee, die begrenzt auf virtuelle Welten angewendet wird. Ob als Grundlage für die Konstruktion von Häusern, als Bauweise im Produktdesign oder als Ansatz für die Produktion von Assetkits in Computerspielen - sie wird vielseitig eingesetzt.

In der Architektur wird auch von „Prefabrication Houses“ geschrieben[2], das heißt, es werden vorgefertigte Häuser gebaut, die aus modularen Einzelteilen bestehen. Ein Beispiel für diese Einzelteile sind die verschiedenen Aluminiumbauteile, die vorproduziert werden:

Es gab hierbei insbesondere auf der Expo 1967 in Montreal das Projekt „Habitat“, bei dem mit vorgefertigten modularen Einheiten ein Wohnkomplex gebaut wurde. Verantwortlich dafür war der Architekt Moshie Safdie, welcher im Alter von 24 auf der Expo ausstellte. Aus den 354 vorgefertigten Einheiten wurden 158 Wohnungen gebaut. Seitdem wurde in der Architektur immer wieder das Thema des modularen, vorgefertigten Hauses thematisiert.[3] Eine Steigerung dessen wurde hierbei durch Werner Aisslingers Loft Cube durchgeführt. Der Loft Cube kann auf Hochhausdächern platziert werden und an Stationen „angedockt“ werden. Wodurch ein modulares Wohnerlebnis möglich wird, was auch Mobilität ermöglicht. Die Produktionszeit nimmt durch die vorgefertigte Bauweise und Festlegung auf bestimmte Größen auf 12 Wochen in Europa ab[4]. Desweiteren wird bei diesem Projekt deutlich, dass durch eine detaillierte Konzeption Kreativität und Individualisierung bei modularen Elementen nicht ausgeschlossen ist. Werden Neubauten in den USA und Deutschland betrachtet, wird schnell deutlich, dass die Fertigbauweise eine große Rolle spielt. In Deutschland basieren 15% aller Neubauten auf der modularen Bauweise. In den USA sind es sogar 90%.[5]

Im Rahmen von vorgefertigten Häusern wird hierbei wird nach denselben Rahmenbedingungen wie auch in der 3D-Grafik gearbeitet. So wird sich wie auch bei der Asset Creation an einem Raster orientiert.

---

[2] Vgl. Ryan, 2010
[3] Vgl. Ryan, 2010
[4] Vgl. Aisslinger, 2014
[5] Vgl. Ullrich, 2006

Wird weiterführend die Gestaltung von architektonischen Elementen betrachtet, finden sich in der Architektur ebenfalls modulare Strukturen wieder. (The Water Cube, Beijing) [modular structures in architecture]

Es wird durch diese Beispiele deutlich, dass in der Architektur die Produktion von Einzelteilen, die Gestaltung der Gebäude sowie architektonische Gestaltungskomponenten Modularität als Lösung für Probleme sowie Gestaltungsmittel angesehen werden.

Modulare Bauweisen gibt es aber nicht nur in der Architektur, sondern auch im Produktdesign. Die Modularität innerhalb von Produktdesign ist an einem mechanischen Stift sehr gut erkennbar[6]:

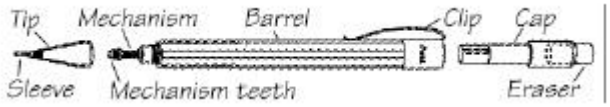

Abbildung 1: Mechanischer Stift

Einzelne Komponenten lassen sich wie auch bei den zuvor gezeigten modularen Häusern austauschen und durch ihre genormte Form variieren. Auf dieselbe Art und Weise ist bereits ein Buch entstanden, bei dem beliebige Hefte von verschiedenen Papiersorten miteinander kombiniert werden können. Auf diese Art und Weise kann man z.B. einen Terminplaner mit einem Skizzenbuch kombinieren. Dieses Konzept nennt sich x17-Konzept und ist hier zu sehen[7]:

Abbildung 2: Modulares Notizbuch

Modularität dient hierbei nicht nur als Sinnbild für Austauschbarkeit von einzelnen Komponenten, sondern auch als Möglichkeit

[6] Vgl. John K. Gershenson, 1999
[7] Vgl. Büttner, 2011

zur Weiterentwicklung eines bestehenden Produktes. Im aktuellen Kontext wird so zurzeit auch ein Androidphone entwickelt, welches durch modulare Bauteile sämtliche Komponenten austauschbar und somit auch upgradefähig werden lässt. Ein Mitglied des Entwicklungsteam macht in einem Interview mit theverge.com deutlich, welches Potenzial das Projekt Ara in sich trägt:

„When this goes to market, this will be the most custom, mass-market product ever created by mankind."[8]

Dabei ist die Besonderheit hervorzuheben, dass nicht nur die einzelnen Bauteile ausgetauscht werden können, wie z.B. CPU, Kamera, Bildschirm usw. sondern auch die Hüllen der Blöcke individualisiert werden können. So will Google anbieten, selbstentworfene 3D-druckfähige Designs online zu bestellen und sich zuschicken zu lassen. Jedes Handy kann somit eigenen Vorstellungen angepasst werden. Jedoch ist dies auch auf die einzelnen Blöcke begrenzt und an das Endoskelett gebunden.

Modulares Produktdesign wirft in diesem Fall auch Probleme auf, denn dort wo Module statt integriertes Design vorherrschen, muss jedes Modul einzeln für sich funktionieren und die Module müssen in das Endoskelett passen und einander verstehen.

Ob das Projekt erfolgreich verlaufen wird, bleibt daher abzuwarten.[9]

Anders als in der Architektur und dem Produktdesign gestaltet sich die Asset Creation in Computerspielen rein digital. Kosten für Materialien entfallen demzufolge und an deren Stelle treten Lizenzgebühren für Soft- und Hardware sowie höhere Personalkosten für Fachkräfte. Jedoch können diese auch mehr produzieren; so gab es bei Skyrim lediglich 2 Kit-Artists[10] Bethesda. Modularität ist hierbei ein Konzept, was bereits seit Super Mario durchgeführt wird[11].

Die Gears-of-War-Serie bildet hierbei keine Ausnahme und wurde innerhalb der Unreal Engine entwickelt. Epic definiert klar den Aufgabenbereich von Environment Artists: „An environment artists goal is to make a seamless robust set of Lego that allows the level designers to quickly shell out a level […]"12

[8] Vgl. Bohn, 2014
[9] Vgl. Wendekind, 2014
[10] Vgl. Burgess, 2013
[11] Vgl. Jones, 2011
[12] Vgl. Epic Games, Inc., 2001-2012

Wie auch bereits bei anderen modularen Ansätzen wird das Raster hierbei als Rahmenbedingung für einen funktionsfähigen modularen Baukasten angesehen.

Hierbei ergibt sich eine Besonderheit: es entstehen bei der Erstellung eines Assetkits verschiedene Regeln, wonach die Gegenstände modelliert werden. Dies wird bei Epic ebenfalls untermauert: „It's likewise important to have a solid workflow that helps you stick to the rules and ensures you are efficient in how you approach a 'set' of environment assets. „

Das Befolgen dieser Regeln, garantiert einen effizienten Workflow und macht den Vorteil der Asset Creation anstelle von Single Mesh Creation aus.

Hierbei sind weitere Optimierungen möglich, wie bereits ein Modder aus der Polycount-Community aufzeigt: Seine Herangehensweise war es anstelle von vielen einzelnen Assets ein Gebäude zu modellieren, zu texturieren und zu mappen. Dieses wird dann wieder variiert und dasselbe erneut wiederholt. So entsteht aus einem einfachen Torbogen ein Haus mit Wohnraum auf Steinsäulen.13

Ein weiterer Fakt, der hierbei anhand eines Mount&Blade-Mods deutlich wird, ist, dass durch die niedrige Auflösung von Assetkits für Computerspiele die Formsprache der 3D-Grafiken beachtet werden müssen. Form follows function14. Bei Low-Poly-Modelling bedeutet dies: So viel wie nötig, so wenig wie möglich.15

Abschließend lässt sich hierbei sagen, dass es unzählige weitere Beispiele für modulares Environment-Design gibt. Die Mass-Effect Reihe von Bioware sei hierbei nur am Rande erwähnt16. Ein Projekt, auf das in dieser Arbeit im Kapitel 2.2 näher eingegangen wird, ist The Elder Scrolls 5: Skyrim.17

Es gibt von der Methode, modulare Bauweisen für Assetkits zu implementieren, verschiedene Ansätze, die explizit für virtuelle Welten, wie Computerspiele, entworfen wurden. Zu den Vertretern gehören hierbei Paul Mader, Lee Perry und Tyler Wanlass.

[13] Vgl. Jelsoft Enterprises Ltd., 2000-2014
[14] Vgl. Greenough, 1947
[15] Vgl. Jelsoft Enterprises Ltd., 2000-2014
[16] Vgl. Jones, 2011
[17] Vgl. Burgess, 2013

2005 schrieb Paul Mader einen Artikel für gamasutra.com in dem er seinen Ansatz für „Creating Modular Game Art For Fast Level Design“ vorstellte[18]. Er hob hierbei zwei Schwerpunkte hervor: 1. Das Raster und 2. den Drehpunkt eines Objektes.

Es gibt hierbei zwei verschiedene Rastereinteilungen. Die erste ist das metrische Raster, das zweite das einheitenbasierte Raster. Auf beide Arten wird in Kapitel 2.2 näher eingegangen. Neben der Bedingung, die Objekte nach dem Raster ausgerichtet zu platzieren, legt Mader insbesondere Wert auf die Platzierung des Drehpunktes. Er nennt hierbei fünf verschiedene Ansätze für die Ausrichtung desselbigen. Variante 1 bezieht sich auf den Mittelpunkt des Objektes. Bei Variante 2 wird der Drehpunkt auf eine Symmetrieachse gelegt. Nummer drei dreht sich um die Ausrichtung des Modells in Bezug zu seiner Umgebung. Steht eine Kiste auf dem Boden, so wird der Mittelpunkt gewählt und auf die untere Standfläche geschoben.[19]

Bei der vierten Variante wird von Tiling models ausgegangen, also Assets die aus verschiedenen Teilen wie gerade Teile und Ecken bestehen. Der Drehpunkt wird nun an eine Kante geschoben, so kann das Objekt beliebig skaliert werden, ändert aber seine Position nur in Bezug auf eine Seite.

Abschließend erklärt Mader die Möglichkeit, den Drehpunkt als Rotationspunkt zu nutzen. Dabei wird der Drehpunkt außerhalb des Modells platziert und ermöglicht somit, zusammen mit einem Grid Snap, ein einfaches Erstellen von Kurven aus einem einfachen Teil ohne Lücken oder Überlappungen.

Lee Perry hatte hierbei bereits 2002 die Ansätze und Denkweise für Modularität im Environment-Design erläutert. Er ist hierbei repräsentativ für Epic Games.

Perry betont hierbei insbesondere die Vorbereitung für die Erstellung eines modularen Assetkits.[20] Bei Epic wird demzufolge erst über die Skalierung des Levels entschieden. Werden für ein Autorennspiel nur große modulare Blöcke, z.B. Gebäude, gebraucht oder aber kleinere detaillierte Teile für z. B. das Interior eines Raumschiffs, wo der Spieler in Ego-Perspektive die Umgebung erkundet. An zweiter Stelle wird über die Proportionen entschieden, wonach das Raster bestimmt wird. Dabei sollten auch Handlungen der Charaktere, wie Sitzen auf einem Stuhl oder die Türrahmengröße, berücksichtigt werden.

[18] Vgl. Mader, 2005
[19] Vgl. Mader, 2005
[20] Vgl. Perry, 2002

Ist die Skalierung und das Raster bestimmt, muss eine Liste mit Asset-Teilen angefertigt werden. Weiterführend muss die Key-Feature-Liste stehen, um das Environment näher zu beleuchten. Auf diese Weise wird deutlich, was für das Grundset benötigt wird und welche Teile, besonders sehr komplexe Geometrie, vorerst zurückgestellt werden können. Auch die Endteile für z.B. Flüsse müssen gut durchdacht werden und für den Spieler Sinn machen, da sonst die Modularität die Kontinuität der Spielwelt stören kann.

Perry nennt zudem „Transitional pieces“, das heißt Übergangsgeometrie, welche es ermöglicht das Assetkit weiterführend kombinierbar zu machen. Jedoch sollten Teile grundsätzlich nach so vielen Seiten wie möglich symmetrisch sein, um so viel wie möglich Kombinationsmöglichkeiten mit anderen Teilen zu bieten. Accessory pieces, also Zubehör zu dem Basisset, auch sogenannte „Props“, sollen ebenfalls durchdacht und entwickelt werden. So kann z.B. Efeu unschönes Mapping oder Übergange an Kombinationsteilen verdecken und das Level um ein Detail bereichern.

Eine Einschränkung an dieser Stelle ist auch die Beschaffenheit des Teams. Für einige Firmen ist es leichter einen Workflow zur Erstellung modularer Assetkits zu erstellen, für andere schwieriger. Perry rundet den Artikel im gd mag mit dem Ausblick auf detailreiche und trotzdem zügig erstellte Level ab.[21]

Tyler Wanlass hatte im Gegenzug, 2011, zu Perry und Mader den Ansatz, nicht mit der Geometrie zu beginnen, sondern erst eine gerasterte Textur und danach die 3D-Geometrie zu erstellen[22]. Das Raster der Textur und des Levels sollten gleich sein und werden nach der Power of 2 (siehe Kapitel 2.2) aufgebaut. Hierbei wird die Textur nach den geometrischen Elementen gegliedert, z.B. Mauersteine, Tür und Fenster.

Im Digital Tutor Tutorial von Joshua Kinney wird dieselbe Logik angewandt. Kinney zeigt zudem, dass so auch Variationen von symmetrischen Gegenständen, z.B. Türen, platzsparend in einer Textur untergebracht werden können[23]. So entwirft er jeweils zwei unterschiedliche Türhälften, die auf der Geometrie gespiegelt zwei unterschiedliche Türen ergeben.

Durch die Methode „nach Textur“ zu arbeiten und mit der Power of 2 kommt es zu einer perfekten Passform in Bezug auf die Pixel Ratio und das Raster.

---

[21] Vgl. Perry, 2002
[22] Vgl. Wanlass 2010-2014
[23] Vgl. Kinney, 2012

Die Herangehensweise an die Modellierung der Assets unterscheidet sich hierbei nicht von Mader oder Perry. Kinney fügt hierbei dem Workflow von Wanlass noch die Idee der „inverted corner“, als Grundidee für Basisserien bei Gebäuden, hinzu.

Zusammenfassend bleiben zwei grundsätzliche Herangehensweisen:

1. Es wird beginnend mit der Geometrie ein Assetkit erstellt.
2. Es wird beginnend mit der Textur ein Assetkit erstellt.

Die Plannung der Assetkits ist hierbei allerdings gleich. So werden die Skalierung, das Raster und eventuelle Props, Accessory Tiles oder Transitional Pieces vor dem Erstellen festgelegt.

Ableitend lässt sich demzufolge sagen, dass Modularität immer die Kombination von mehreren Modulen, also Assets zu einem großen Ganzen beschreibt. Diese sollen durch ihre Wiederverwendbarkeit einen finanzellen, zeitlichen und qualitativen Mehrwert schaffen. Epic Games hat hierzu 2011 folgendes gesagt:

“modular design is concerned with making lots of high-quality chunks of levels and reusing those chunks intelligently”

Demzufolge minimieren Assetkits auf lange Sicht den Zeitaufwand, um Level zu erstellen, wodurch Produktionsspielraum für weitere Props oder Animationen usw. entsteht. Assetkits begünstigen hierbei auch die Konsistenz im Design – also die Qualität der virtuellen Welt und sind durch ihren unbegrenzten erneuten Einsatz auch auf andere Projekte übertragbar und können somit Kosten sparen.

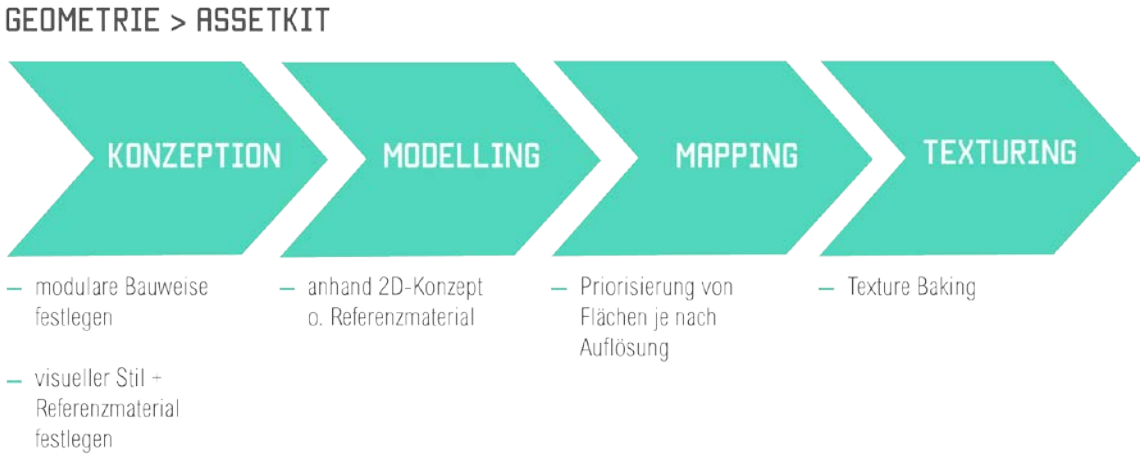

Abbildung 3: Assetkiterstellung nach Geometrie

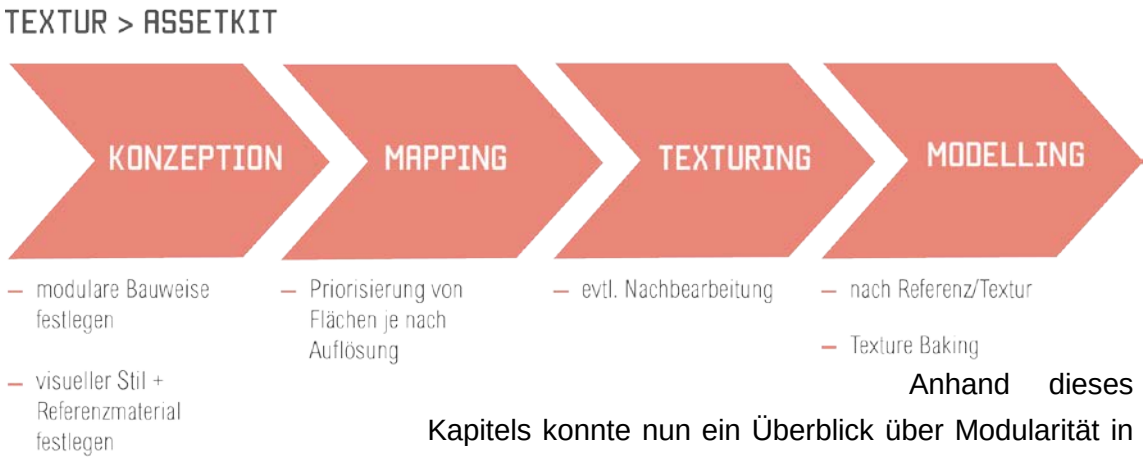

Abbildung 4: Assetkiterstellung nach Textur

Anhand dieses Kapitels konnte nun ein Überblick über Modularität in unserer alltäglichen Welt sowie der Bezug zu Modularität im 3D-Design gegeben werden. Hierbei konnte bereits durch die Vertreter Wanlass, Perry, Mader und Kinney aufgezeigt werden, dass es verschiedene Herangehensweisen und Schwerpunkte innerhalb der Workflows gab. Es stellen sich hierbei also für die weiteren Betrachtungen dieser Arbeit zwei grundlegende Fragen:

1. Inwiefern hat sich Modular Asset Creation im aktuellen Kontext weiterentwickelt?

2. Wie muss ein Konzept zur Erstellung eines Assetkits aussehen?

Für die Beantwortung der ersten Frage wird im nächsten Kapitel der Workflow von Bethesda detailliert betrachtet.

2.2 Power of 2

Im fünften Teil der Elder Scrolls Reihe, The Elder Scrolls: Skyrim, werden modulare Assetkits verwendet. In erster Linie steht hierbei die Frage im Raum, warum gerade Bethesda auf Modularität zurückgreift. Joel Burgess, ein Entwickler der an Skyrim mitgearbeitet hat, bringt es in seiner GDC 2013 Präsentation auf den Punkt: „Bethesda Games are big“[24].

[24] Vgl. Burgess, 2013

Mader führte hierbei bzgl. des Power-of-2-Systems folgendes aus[25]: Das System basiert auf der Zahl 2, wie z.B. die Texturauflösungen (512x512, 1024y1024 [..], Anm. d. Verf.) und optimiert auf diese Weise den Workflow. Hierbei ist definiert, dass 53 Einheiten einen Meter darstellen und ein Charakter laut Epic Games somit 96 Einheiten, also 1,80m, hoch ist. Hierbei steht das System im Zusammenhang mit dem Raster:

2^x = Grid size
2^0 = 1 Unit
2^1 = 2 Units
2^2 = 4 Units
2^3 = 8 Units
2^8 = 256 Units
2^{10} = 1024 Units

Abbildung 5: Einheitensystem

In seinem Blog fasst er dies erneut zusammen: „The modular approach to level design we're analyzing here is just one manifestation of the BGS studio culture."26

Das folgende Kapitel bezieht sich auf die Ausführungen von Joel Burgess bei der GDC 2013.

In Skyrim musste folgendes Environment erstellt werden:

- 16 sq. mile Overworld
- 5 Major Cities
- 2 Hidden Worldspaces
- 300+ Dungeons
- 140+ Points of Interest
- 37 Towns, Farms & Villages

Dies wäre ohne Assetkits nicht möglich gewesen. Es arbeiteten insgesamt nur 90 Entwickler an Skyrim.

Das Besondere innerhalb des Skyrim-Workflows ist hierbei, dass auf dem Power of 2 System aufgebaut wird und Modularität auf einem weiteren Level aufgebrochen wird.

[25] Vgl. Mader, 2005
[26] Vgl. Burgess, 2013

So wurde das Zwergen-Assetkit mit Outerior Teilen kombiniert oder ein Eishöhlenkit mit ihm verschmolzen. Auf diese Art und Weise bleibt das Setting visuell ansprechend und wird aufgelockert. Die Mehrfachverwendung bleibt somit weiterhin bestehen. Diese Vorgehensweise nennt man auch kitbashing.

Abbildung 6: Kitbashing

Bei Bethesda werden Assetkits durch einen festgelegten Prozess erstellt. Zuerst beginnt die Konzeptionsphase. In der entscheidet sich welche Hauptidee hinter dem Kit steht. Es muss festgestellt werden, wo das Kit innerhalb des Spieles platziert werden soll. Daher stellen sich Fragen, wie: Was ist das visuelle Thema des Kits? Wie unterscheidet es sich zu anderen innerhalb des Spieles? Was soll damit gebaut werden? Wie wird es genutzt? Wie oft wird es genutzt?

Je nachdem, wie häufig ein Kit in einem Spiel verwendet wird, umso größer bzw. kleiner ist sein Umfang innerhalb des Spieles. Für Assetkits kann es zudem sogenannte Sub-Kits geben, d.h. die Teile eines Assetkits, die mit allen anderen kombinierbar sind aber in sich ein geschlossenes System bilden. Als Beispiel wird hierbei z.B. ein small hallway subkit aufgeführt, d.h. modulare Bauteile um einen Weg zu erstellen.

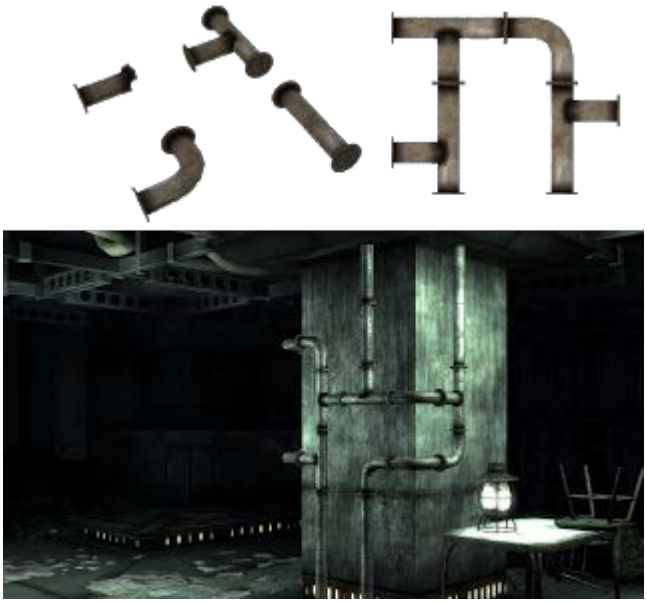

Abbildung 7: Pipekit (Subkit)

Auf Basis dieser Planungen kann entschieden werden, wie viel Zeit für ein Kit eingeplant werden muss.

Als zweite Phase gibt Bethesda die Belegphase an. Dabei wird die Funktion der Konzeptideen überprüft und es werden grobe Bauteile der Assetserie erstellt. Diese haben keine geometrischen Details oder Texturen. Proportionen, Logik des Kits und Namensgebungen werden ausgetestet. Bethesda veranschlagt dafür 1-3 Wochen.

Bevor Assets erstellt werden, wird über den sog. Footprint des Kits entschieden, d.h. das Grundprofil eines Kits. Bethesda betont, dass die meisten Kits gleichseitig aufgebaut sind, allerdings kann eine anders gewichtete Proportion (z.B. rechteckig, Anm. d. Verf.) ein individuelleres Gefühl vermitteln.

Diese Grundentscheidungen beeinflussen ebenfalls den visuellen Stil des Kits. Zusätzlich wird entschieden welche Snap Sizes, also Abstände zum Einschnappen des Rasters, festgelegt werden. Große Abstände begünstigen hierbei einen flotten Workflow für Leveldesigner. Ein weiterer Punkt ist die axiale Symmetrie eines Assetkits. Es wird hierbei von Bethesda vermieden Teile zu bauen die auf allen drei Achsen symmetrisch angelegt werden müssen. Eher werden die Kitbauteile aufgeteilt. So entsteht z.B. ein Weg-Kit, das nur auf der X-Achse symmetrisch ist. Der Grund hierfür ist, dass auf jeder Achse symmetrische Assetkits schwer zu erstellen sind. Trotzdem wird betont, dass sie größeren kreativen Freiraum beim Erschaffen bieten.

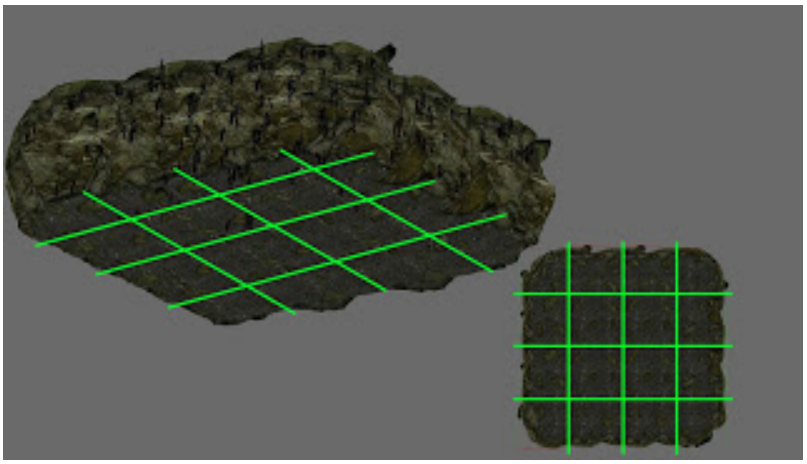

Abbildung 8: Footprint

Desweiteren wird ebenfalls verdeutlicht, dass die äußeren Grenzen des Footprints (Siehe Abbildung7, Anm. d. Verf.) beachtet werden müssen, damit es keine Grafikfehler gibt. Ein Beispiel ist hierfür das Hallway-Kit. Bei dem ersten Entwurf keine Wanddicke bedacht wird und somit planare Ebenen die Wände trennen.

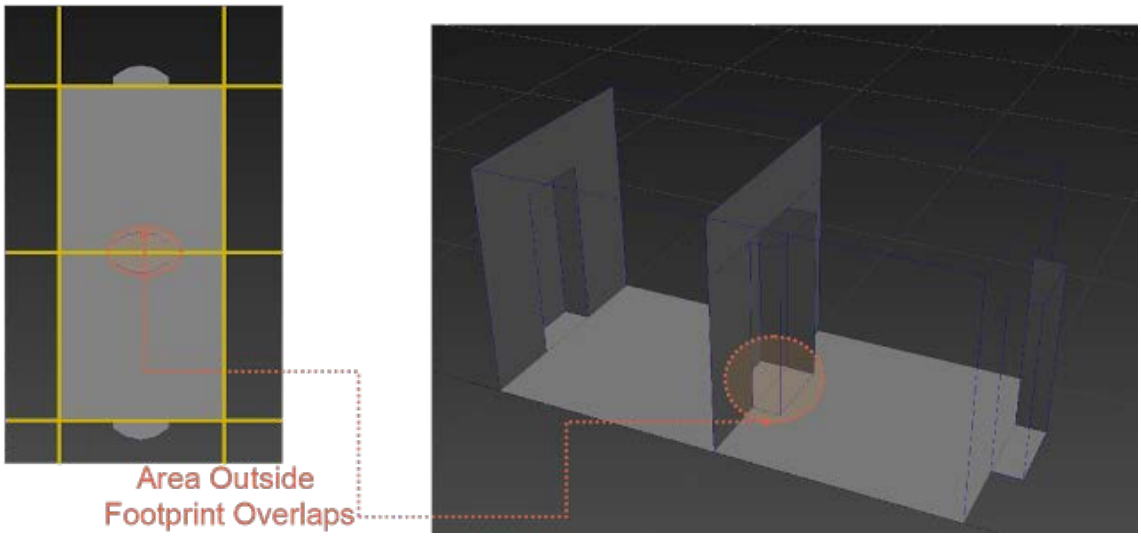

Abbildung 9: Footprint mit Überlappungen

Bei dem zweiten Entwurf wird den Durchgangsnischen der Türen Platz geboten und somit eine Wanddicke assoziiert.

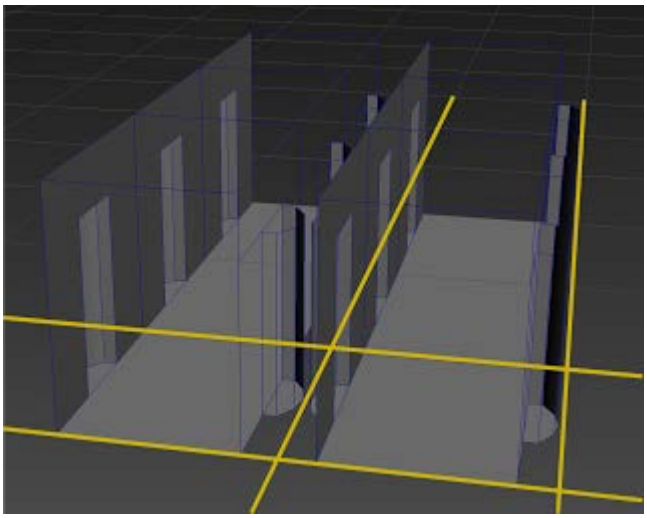

Abbildung 10: Footprint-Korrektur

Es ist wichtig solche Fehler zu Beginn zu finden und im späteren Produktionsprozess somit Zeit und Kosten zu sparen.

Als abschließender Schritt der Belegphase in Bethesdas Kit-Entwicklungsprozess ist ein Stresstest des Kits durch den Leveldesigner. Es werden verschiedene Szenarien ausgetestet, um die Grenzen des Assetkits auszuloten. So wird ein Hallway-Kit geloopt, damit man die Funktionalität der Ecken testen kann. Als Nächstes wird das Assetkit vertikal gestapelt, wodurch klar wird, ob Regeln für die Wand-/Deckendicke eingeführt worden sind. Den letzten Schritt bilden Szenarien, bei denen die Kit-Teile nicht wie beabsichtigt, sondern flexibel eingesetzt werden. So wird z.B. eine Hallway-Ecke genutzt, um Säulen zu erstellen.

Anhand dieses Vorgehens wird deutlich, an welchen Stellen das Kit verbessert werden muss und wo Grenzen sichtbar werden. Dieser Prozess findet mit ständiger Rücksprache von Kit-Designer und Leveldesigner statt. Somit entsteht ein Kit, welches visuell ansprechend und auch gut spielbar ist.

Die dritte Phase nennt sich Graybox-Phase (vergleichbar mit einer detaillierten Blocking-Out-Phase, in der Geometrie erstmals segmentiert ist und das Level nicht nur aus Grundkörpern zusammengesetzt ist, Anm. d. Verf.).

Abbildung 11: Graybox-Assets

Sie dient dazu die Basisteile des Kits festzulegen. Hierfür werden die Basisteile, angepasst auf das Thema visuell um Kernelemente erweitert. Wird z.B. ein großes Rohr benötigt, welches in der Mitte eines Raumes platziert ist und über mehrere Teile symmetrisch bzw. kombinierbar sein soll, wird dieses nun eingebaut. Ein weiteres Beispiel wäre asymmetrische Höhlengeometrie. Wichtig ist, dass die Graybox nicht zu generisch sein darf, weil so die speziellen Fälle für die Kits nicht ausgetestet werden können.

Gemeinsam mit dem Leveldesigner wird zudem die Namensgebung und die Pivot-Point Platzierung besprochen. Die gesamte Phase dauert bei Bethesda 1-4 Wochen.

Als vierte Phase ist die Erstellungsphase vorgesehen, in welcher das Kit gebaut und texturiert wird. In diesem Zeitraum werden mehrere Leveldesigner für neue Stresstests benötigt und somit weiteres Feedback generiert. Je nachdem wie viele Subkits eingearbeitet werden müssen, dauert die Erstellung mehrere Monate. Sogenannte Hero Pieces, d.h. visuell komplexe Einzelstücke wie kunstvolle Statuen in Säulen usw. sollen laut Bethesda erst zu einem späteren Zeitpunkt erstellt werden, da die größte Zeit den Assets gilt, die oft im Spiel gesehen werden, also den Standard-Assets. Es wird ebenfalls angeraten Help Markers zu setzen, wodurch, wie in Abbildung 11 zu sehen ist, die Grenze von Assets besser abgeschätzt werden kann.

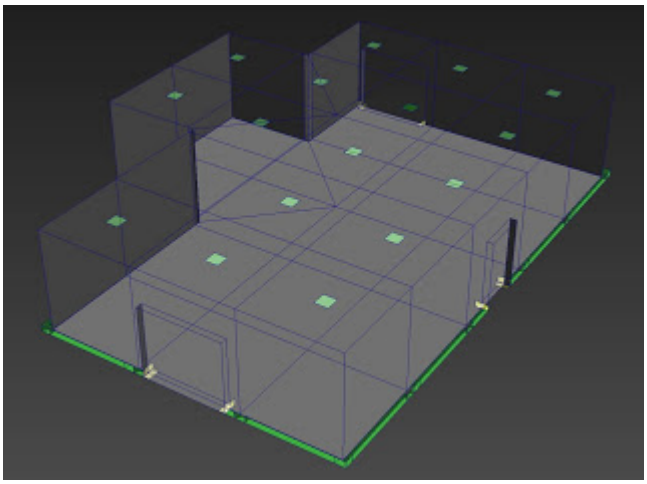

Abbildung 12: Help-Markers

An fünfter Stelle steht die Polish Phase, bei der über mehrere Monate Bugs gefixt werden und nach Bedarf die Funktionalität des Kits erweitert wird. Es gilt hierbei auch einzuschätzen, ob alle Anfragen diesbezüglich bearbeitet werden müssen, oder ob das Kit dadurch nur aufgebläht wird.

Hierbei gilt es auch die typische Rasterstruktur des Assetkits ein Stück weit aufzubrechen. Dies gelingt mit dem sog. "Pivot and Flange"-Systems, bei dem Assets nicht nur um den eigenen Pivot gedreht werden, sondern um ein ausgewähltes Referenzobjekt herum, wodurch neue Winkel möglich werden. Dadurch können allerdings beim Levelbau auch Lücken oder Überlappungen entstehen, die mit neuen Teilen ausgeglichen werden müssen.

Eine weitere Herausforderung bilden Höhlenkits, die von den Bethesda-Designern durch ein Shell-Based-Building-System ausgeglichen wurden. D.h. es wurde Wandgeometrie innerhalb der Ecken platziert und durch Säulen und Vorbauten ergänzt, um eine glaubhafte organische Struktur darzustellen. Burgess ergänzt hierbei noch die Wichtigkeit von guter Ausleuchtung, die kleine Fehler verbergen.

Bethesda unterscheidet hierbei zwischen richtungsbegrenzten Kits und traditionellen Kits. Richtungsbegrenzte Kits haben zudem Assets, die zueinander in Beziehung stehen und ohne die sie nicht verwendet werden können, garantieren aber auch Flexibilität in der visuellen Gestaltung. So ist ein asymmetrischer Gehweg möglich.

Abbildung 13: Asymmetrischer Gehweg

Insbesondere bei organischen Kits ist diese Variationsvielfalt sehr vorteilhaft - bedeutet allerdings auch mehr Arbeit für das Art- und Designteam. Für architektonische Kits machen richtungsbegrenzte Kits keinen Sinn, da sie zuviel Aufwand bedeuten.

Ein weiteres Konzept, was Bethesda hierbei etablierte, ist das "De-Twist"-Prinzip. Hierbei gibt es durch die Kombination von A- und B-Seiten des Durchgangs die Möglichkeit den Weg zu loopen.

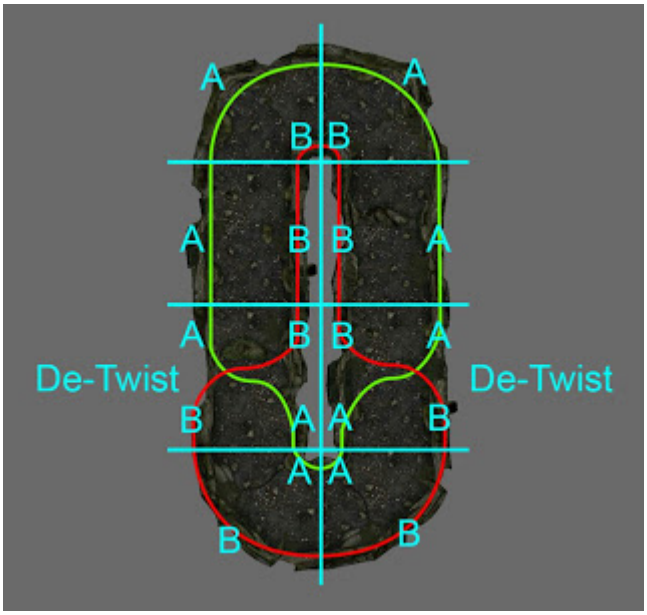

Abbildung 14: De-Twist-Prinzip

Das "Platform"-Kit dient dazu das Level horizontal abzuschneiden und die Teile dessen werden stets auf Oberflächen gesetzt. Bethesda nennt dieses Kits auch "Glue" Kits, wie Perry zuvor die Transitional Pieces. Sie dienen dazu Übergänge zwischen ver-

schiedenen Kits zu schaffen. Als Beispiel können Tunnel, Plattformen oder Leisten aufgeführt werden.

Zusammenfassend ist somit nun der Kit-Building-Process von verschiedenen Seiten beleuchtet worden, sowohl historisch durch Paul Mader, Lee Perry und Tyler Wanlass, als auch der detaillierte aktuelle Stand nach einer Power-of-2-Einteilung von Bethesda.

Abbildung 15: Workflow Bethesda

Im Laufe dieses Kapitels wurde gezeigt, dass der Bethesda Workflow mehrere Stufen durchläuft und insbesondere der gestaltungstechnische Aspekt näher in den Vordergrund gerückt wird. Die reine Funktionalität, die noch bei Mader, Perry und Wanlass im Vordergrund stand wurde nun durch explizite Tests und Szenarios definiert. Hinzu kommt ebenfalls die Option des Kitbashings, welche Bethesda nutzt und somit die Wiederverwendbarkeit von Assetkits erhöht.

Zudem wird für die Frage „Wie muss ein Konzept zur Erstellung eines Assetkits aussehen?“ eine erste Teilantwort deutlich:

Die Konzeptionsphase spielt eine große Rolle. Bevor das Assetkits gestalterisch umgesetzt wird muss erst die Funktionalität anhand von ausgewählten Szenarios gewährleistet werden.

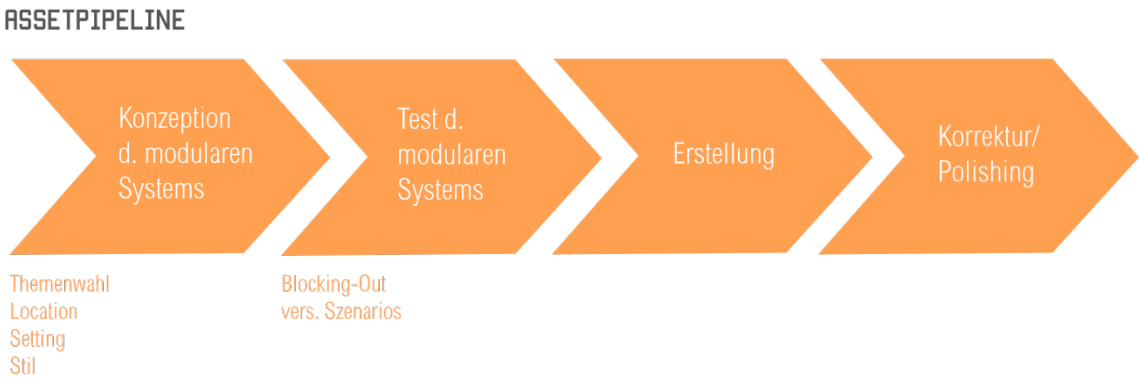

Abbildung 16: Allgemeine Schlussfolgerungen zur Assetpipeline

Es stellt sich nun die Frage, wie von einem Power-of-2-Einheitensystem von Bethesda ein Bezug zu realen Orten geschaffen werden kann. Innerhalb des Praxisprojektes soll eine reale Location in Form eines virtuellen Filmsets umgesetzt werden. Das Thema ist hierbei „Haus im Wald“ - dieses basiert auf dem metrischen System.

Die Forschungsgruppe Gamecast hat hierbei das Rapidbox-System entwickelt und liefert somit eine potenzielle Antwort auf diese Frage. Anhand der vordefinierten Szene soll zudem der Test erfolgen, ob das Rapidbox-System mit seiner Bauweise bei der Umsetzung erfolgreich ist.

Im nächsten Kapitel gilt es nun die Bedingungen, unter denen das Assetkit entwickelt wird, zu beleuchten. Welchen Ansatz verfolgt das verwendete Rapidbox-System? Ermöglicht es die Umsetzung des genannten Themas durch seine Definition?

2.3 Rapidbox

Bei der Rapidbox-Bauweise handelt es sich um ein metrisches Assetkit, welches festgelegte Grundteile besitzt. Diese bilden die sogenannte Basisserie. Diese Bauweise ist

hierbei ein von der Forschungsgruppe Gamecast entwickelter Standard und kein genereller Gamestandard wie Power-of-2. Er befindet sich in der Entwicklung und dient zur allgemeingültigen Bildung von themenorientierten Assetkits. Modulare Assetkits, die sonst wie bei Skyrim[27]individuell angepasst werden sind hier durch den Rapidboxstandard genau definiert.[28] Für einen Flugsimulator basiert das Assetkit so z.B. auf kompletten Gebäuden in der Außenansicht, bei einem Spiel in der Ego-Perspektive allerallerdings auf Einzelteile von Häusern.

Durch die Standardisierung von Rapidbox entsteht der Vorteil, dass das Assetkit erweiterbar ist ohne dessen Grundstruktur zu verändern und beliebig viele Themen und Kombinationen möglich sind. Jedoch heißt dies auch, dass die Gestaltungsfreiheit begrenzt ist, da die Grundstruktur die Variation an Teilen verringert. Zudem wird ein Assetmanagementsystem ab einer gewissen Anzahl an Themen notwendig.

Im Mittelpunkt steht hierbei nach architektonischen Maßen arbeiten zu können und ein Gebäude ohne Umrechnung in beliebiger Länge bauen zu können.

Aufgrund dessen teilt sich das Basiskit in Grundteile wie Base, Middle und High. Es wird also horizontal und vertikel kategorisiert. Bei den Wandelementen gibt es somit Low-, Middle- und High-Assets. Eine Wand mit der Kategorie High und z.B. der Größe 4x4m wäre somit WHS4_1, wobei das S für Side steht und _1 für die Variante. So ist es möglich optische Variationen zu erstellen und einfach durchzunummerieren.

[27] Vgl. Burgess, 2013
[28] Vgl. Cinector, 2013

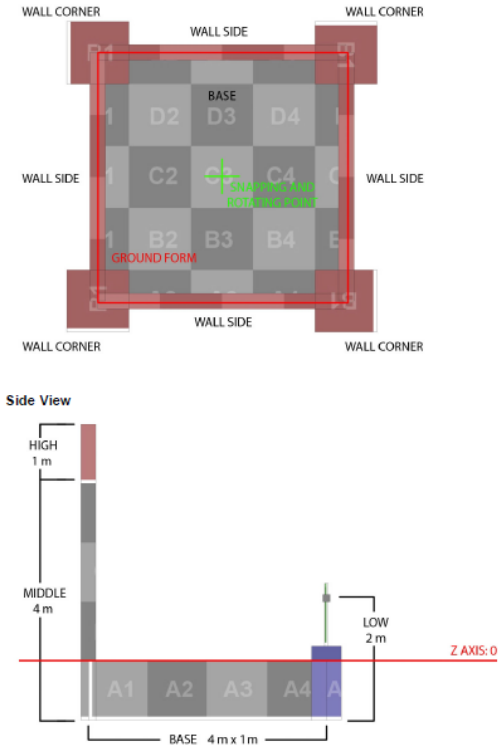

Abbildung 17: Rapidbox-Aufbauprinzip

Selbiges gilt für die Base-Elemente, welche durch die Kategorien Roof, Ceiling oder Floor näher beschrieben werden.

Hierbei ein Einblick in die Basisserie, siehe Abbildung 16 und eine Testszene mit dem Mittelalterset.

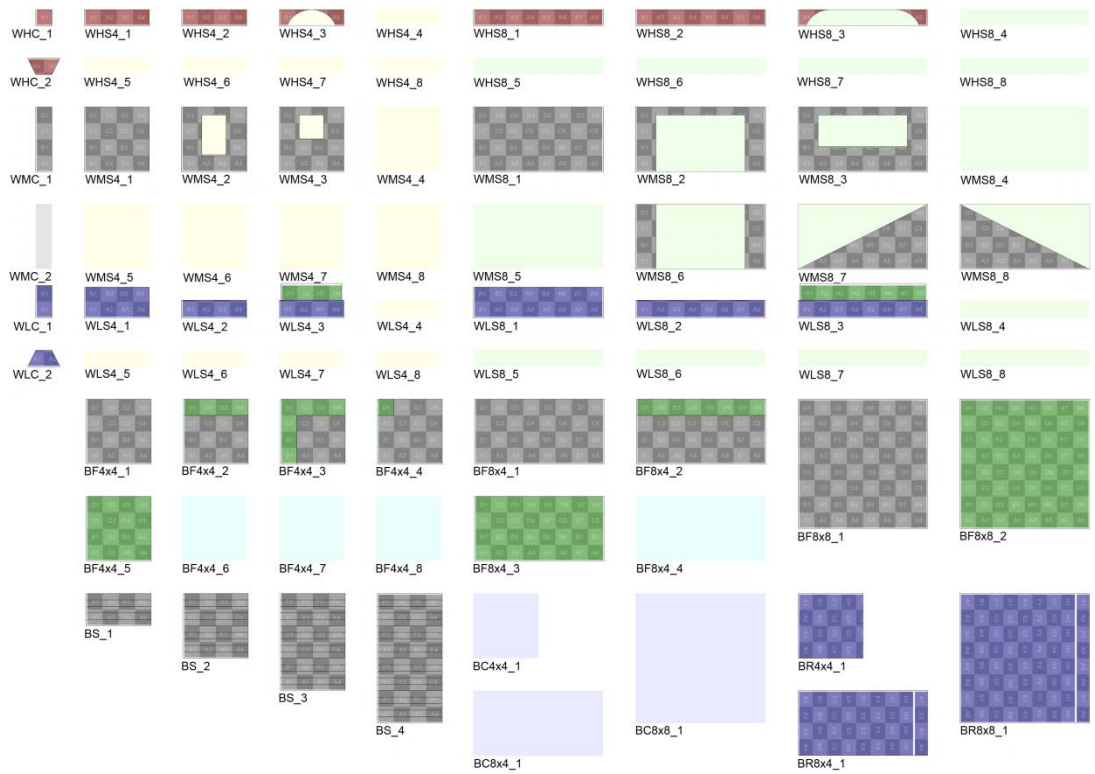

Abbildung 18: Basisserie von Rapidbox

Abbildung 19: Mittelalterset Rapidbox

Es wird deutlich, dass diese Teile ebenfalls an einem festen Raster ausgerichtet werden müssen. Das Grid Snapping sollte auf 1 oder 4m eingestellt sein, damit die Teile direkt aneinander anliegen bzw. ein gutes Arbeiten möglich ist. In der Vogelperspektive zu arbeiten, erleichtert zudem die Platzierung der Grundfläche.

Auf Basis dieses Systems ist eine realistische Abbildung von realen Filmsets möglich. Zudem kann auf Basis der Namensgebung ein Set ausgetauscht werden. Dies wird in Kapitel 5 näher erläutert und macht es möglich als Regisseur flexibel zu arbeiten. Ein Assetkit ist somit nicht spezifisch definiert wie ein Skyrim-Assetkit, sondern auf die Basis heruntergebrochen und dadurch allgemeiner einsetzbar.

Zu vergleichen ist dies insbesondere mit Lego. D.h. es werden Sets entwickelt wie z.B. das Piratenset von Lego, das auf den Grundbausteinen des Lego-Systems basiert. Diese Teile sind universell und können für die Generierung immer neuer Sets genutzt werden.[29]

Ein Nachteil der hieraus entspringt ist der repetitive Levelcharakter (Anm. d. Verf., d.h. Wirkung d. Levels). Es ist also eine gewisse Assetanzahl notwendig, um den modularen Charakter aufzubrechen, sowie der Einsatz von Props vorgesehen. Die Basisserie wird zudem von der Forschungsgruppe mit der Zeit erweitert, wodurch mehrere Variationen entstehen.

Abschließend wird somit deutlich, dass das Rapidbox-System mit seiner allgemeineren Bauweise Assetkits austauschbar werden lassen will und somit die Variation jeden beliebigen Themas möglich macht.

Es bringt hierbei das Rüstzeug mit, um ein beliebiges Thema, wie auch „Haus im Wald“ umzusetzen. Dies muss zudem in der Umsetzung anhand eines Grundrisses näher geprüft werden.

Doch in welchem Rahmen ist dies gegeben? Welche technischen Grenzen bestehen hierbei? Welche Bedingungen werden hierbei an ein Asset gestellt?

Dies gilt es im nächsten Kapitel zu verfolgen.

[29] Vgl. LEGO Group, 2014

3 Technische Begrenzung

3.1 Asset Life Cycle

Wie bereits in den verschiedenen aufgezeigten Workflows deutlich wurde, muss ein Asset bestimmte Bedingungen erfüllen, damit es mit der modularen Bauweise funktioniert. Es sollte in erster Linie Engine-orientiert aufgebaut sein, d.h. Tris oder Quads aufweisen und je nachdem, ob zweiseitige Texturen unterstützt werden oder nicht, zusätzliche Polygone mit der Textur haben.

Auch müssen sich Assets an die vorgegebenen Basisbegrenzungen halten und abgeschlossene Objekte sein, d.h. keine ungenutzten bzw. nichtverschmolzenen Vertices. Ein weiterer Punkt sind Flächen, die in andere hineinragen. Generell muss eine saubere Topologie gegeben sein. Saubere Topologie bedeutet hierbei, dass ein Polygonobjekt nur aus Quads und Tris besteht und kein N-Gon aufweist. Zudem sollte darauf geachtet werden, dass der Polygonfluss – also die Richtung in der die Geometrie aufgebaut ist – schlüssig ist. (Siehe Abbildung 19 und 20, Anm. d. Verf.)

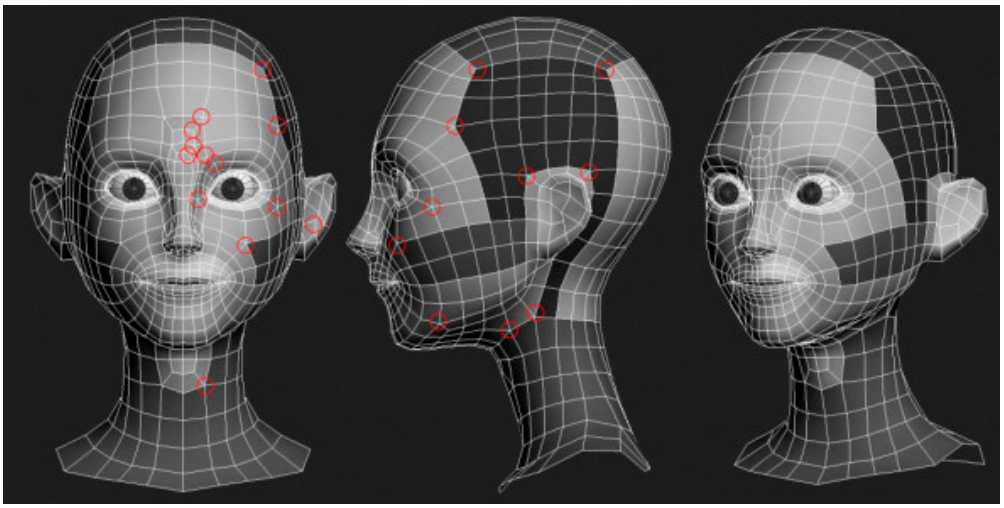

Abbildung 20: Gute Topologie

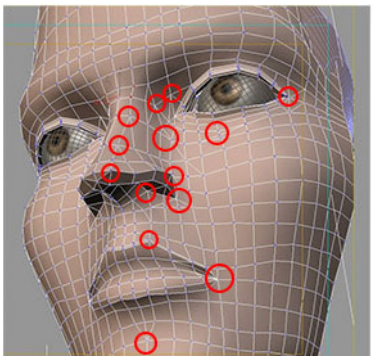

Abbildung 21: Schlechte Topologie

Je nach Verwendungszweck sollte auch die Polygonanzahl eine Rolle spielen. Im Falle der Rapidbox-Bauweise müssen die Assets Low-Poly sein.

Die Assets werden zudem Rapidbox-spezifisch bezeichnet und so auch in die Ordnerstruktur eingepflegt.

Jegliche Assets besitzen zudem mind. eine Diffusemap, Normalmap mit umgekehrten Y-Kanal und Specularmap. Glossymaps können ebenfalls hinzugefügt werden. Je nach Bedarf kann eine Illuminationmap oder Opacitymaske hinzugefügt werden.

Diese Texturen werden geschachtelt.

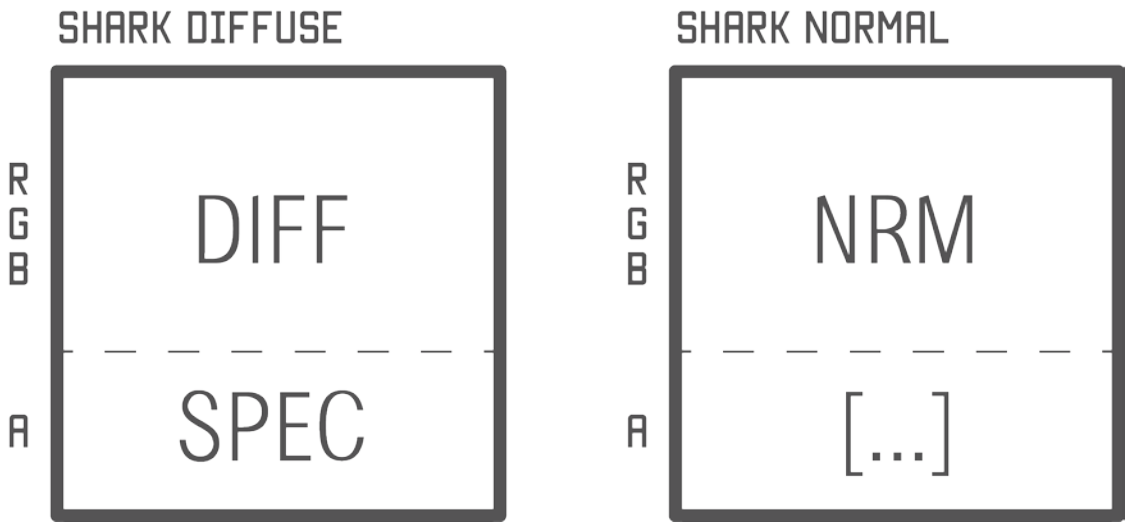

Abbildung 22: Texturimplementierung in die Shark-Engine

Der Pivot-Point des Assets muss ebenfalls je nach Verwendung gesetzt werden. (siehe Kapitel 2.1 Anm. d. Verf.) Er dient hierbei in dem Level als Nullpunkt für das Objekt und bestimmt die Position dessen. Am Ende wird noch das Kollisionsmodell hinzugefügt. Dies bedeutet, das physikalische Volumen eines Körpers zu definieren, worauf z.B. der Spielcharakter reagieren kann. Sei es eine Wand, die mit einer gleich großen und breiten Box definiert wurde, welche der Spieler nicht durchlaufen kann, oder aber ein Schalter in Form einer Bodenplatte, bei dessen Berührung ein Event ausgelöst wird (z.B. eine Tür öffnet sich).

Es kommt nicht nur darauf an, was ein Asset für Bedingungen erfüllt, sondern auch wie es innerhalb der Engine organisiert wird. Innerhalb der Shark3D-Engine gibt es bisher kein Asset Management System in Form eines Browsers. Es erfolgt ein Asssetmanagement über eine Filestruktur und SVN-System.[30] Im Laufe der weiteren Entwicklung ist dies allerdings nicht ausgeschlossen.

Werden andere Engines bzw. die Software Shotgun als Vergleich herangezogen, z.B. das Unreal Development Kit, existiert ein Content Browser innerhalb der Software. In diesem werden die einzelnen Assets organisiert und zugehörige Texturen angezeigt. Zudem können Gruppen erstellt werden und Assetsets zugewiesen werden.

Es erfolgt also ein Asset Management in Form von Gruppen und Packages. Zudem müssen die Assets in das UDK-Format importiert werden. Wird ein Assetpackage (Assetkit, Anm. d. Verf.) genutzt, kann dies aus verschiedenen Gruppen bestehen, z.B. Gruppe 1: Wände, Gruppe 2: Türen und Gruppe 3: Waffen. Danach wird bei den einzelnen Objekten der Asset-Typ festgelegt, wie Mesh, Sound and Music, Texture, Movies and GUI Elements und Material.[31] Je nach Typ ist auch das Format vorbestimmt.

In Unity ist dies noch simpler gelöst, da man dort nur die Ordnerstruktur hat und Assets lediglich in die Unity Project Directory kopiert werden müssen. Es gibt kein spezielles Unity-Format. Dies macht den Import der Assets ebenfalls sehr einfach, da selbst 3ds Max Files gelesen werden können.[32] In der CryEngine werden Assets ebenfalls mit einem engine-eigenen Format importiert. Organisiert sind sie innerhalb des Browser Panels, wo die Ordnerstruktur abgebildet ist. Es gleicht also Unity. Dabei wird in ver-

[30] Vgl. wikipedia, 2014
[31] Vgl. Thron, 2011
[32] Vgl. Seric, 2014

schiedene Asset-Typen unterteilt[33] z.B. Crytek Geometry Format, kurz: .cgf, Character mit .chr und Skinned Render Mesh .skin. Selbiges erfolgt bei den Materialien und Animation Assets, Facial Editor, Audio Assets und Flash Assets.

Eine erweiterte Form von Asset Management System bildet hierbei Shotgun, welches z.B. zur Organisation von Animationsfilmproduktionen genutzt werden kann (oder jeglichen anderen CGI-Inhalten, Anm. d. Verf.). Dieses Tool bietet nicht nur einen Browser, um zwischen 3D-Assets auszuwählen, zu gruppieren oder ganze Szenen abzuspielen, sondern auch die Arbeitsabläufe zu planen. Dazu können Teams eingeteilt und Aufgaben zugewiesen werden. Zudem können Arbeitsabläufe getrackt werden und bei z.B. Reviews Notizen eingefügt werden. Es ist ebenfalls möglich die Aufgaben zu labeln und z.B. "Überfällig" zu markieren. Jedes Teammitglied bekommt hierfür seine Tasks angezeigt und hat eine Inbox, die zusätzliche Mail- oder Chatprogramme überflüssig macht. Eine Integration in die gängigen Tools wie Maya, Photoshop und Nuke ist ebenfalls vorhanden.link[34]

Tabelle 1: Vergleich Asset-Management-Varianten

	SHARK	CRYENGINE	UNITY	SHOTGUN	UDK
ASSET-ORGANISATION	Filestruktur + SVN	Ordner-struktur	Ordner-struktur	Browser	Ordnerstruktur
EIGENES FORMAT	nein	ja	nein	ja	ja
MANAGE-MENT-TOOL	nein	nein	nein	ja	nein
INTEGRATION	nein	nein	nein	ja	nein

Im Rahmen dieses Kapitels wurde deutlich, dass ein Asset in erster Linie in seinem geometrischen Aufbau enginespezifisch aufbereitet werden muss.

[33] Vgl. Johnson, 2014
[34] Vgl. Shotgun Software Inc.

D.h. Polygonart, Pivotpoint-Platzierung, Texturverschachtelung und Kollisionsmodell müssen bestimmt und umgesetzt werden. Im Falle des Rapidbox-Systems muss ebenfalls die vorgegebene Namensstruktur angenommen werden. Je nach Szenengröße ist zudem zu entscheiden wie hoch das „Level of Detail“, also die Polygonanzahl, ausfällt.

Die vorgestellten Assetmanagementsysteme zeigen zudem mögliche Ansätze für die zukünftige Assetorganisation innerhalb von Cinector an. Es entsteht also die Frage wie Assets bzw. Assetkits innerhalb der Software zukünftig strukturiert werden könnten. Dieser Frage wird im Kapitel 6 nachgegangen.

Um Assets zu erstellen wird unterschiedliche Software benötigt, welche die zuvor an das Asset gestellte Bedingungen ermöglichen muss (Texturerstellung und -verschachtelung, Kollisionsmodellerstellung, allgemein die Erstellung dreidimensionaler Geometrie). In den Kapiteln 3.2 und 3.3 wird hierzu projektbezogen Stellung genommen.

3.2 3ds Max / ZBrush + Plugins

Um die Assets zu bauen wird 3D-Software benötigt. Hierbei können sowohl Freeware, wie Blender, als auch kostenpflichtige 3D-Packages, wie Maxons Cinema4D oder Autodesks Maya oder 3ds Max genutzt werden.

Letztere haben hierbei ein größeres Featurespektrum. Aufgrund von Vorerfahrung wird bei dem Praxisbeleg mit der kostenlosen Studentenversion von 3ds Max gearbeitet.

Mit Hilfe von 3ds Max wird somit das sog. Basemesh erstellt, das heißt eine Basisgeometrie für weitere Detaillierungen. Diese Detaillierungen werden in einem 2.5D Zeichenprogramm namens ZBrush vorgenommen[35].

Mit Hilfe dessen kann unabhängig von der Polygonanzahl wie mit Lehm modelliert werden. Dies begünstigt insbesondere organische Strukturen, da mit 3ds Max das Modell nur durch seine geometrischen Flächen bestimmt wird. Mit viereckigen Flächen ist ein Faltenwurf z.B. wesentlich schwieriger darzustellen, als mit digitalem Lehm.

[35] Vgl. Keller, 2011

Innerhalb von Pixologics ZBrush wird das Mesh vielfach unterteilt und ermöglicht so mit, durch Polygonanzahlen bis in den Millionenbereich, das modellieren von Details wie Kratzern, Dellen oder sonstigen Oberflächenveränderungen. Mit dem Werkzeug Dynamesh kann das nun hochaufgelöste Mesh wieder in seiner Polygonanzahl verringert werden. Auf diese Art und Weise ist es möglich, eine niedrigaufgelöste Version für Games bzw. Prävisualisierung herzustellen.

Je nach Workflow kann auch ein High- und Lowpoly-Mesh erstellt werden und dann das Highpoly- auf das Lowpoly für die Normal gebacken werden. Hierbei wird die Normalenstruktur des High-Polys in einer Textur gespeichert und diese dann dem Low-Poly-Mesh zugewiesen.

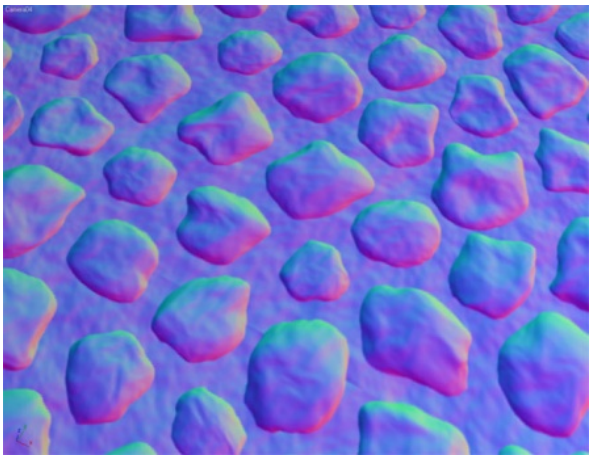

Abbildung 23: Normalmap

Damit dies ohne Fehler passieren kann muss das niedrigaufgelöste Mesh gemappt sein, das heißt die UV-Koordinaten müssen bestimmt worden sein.

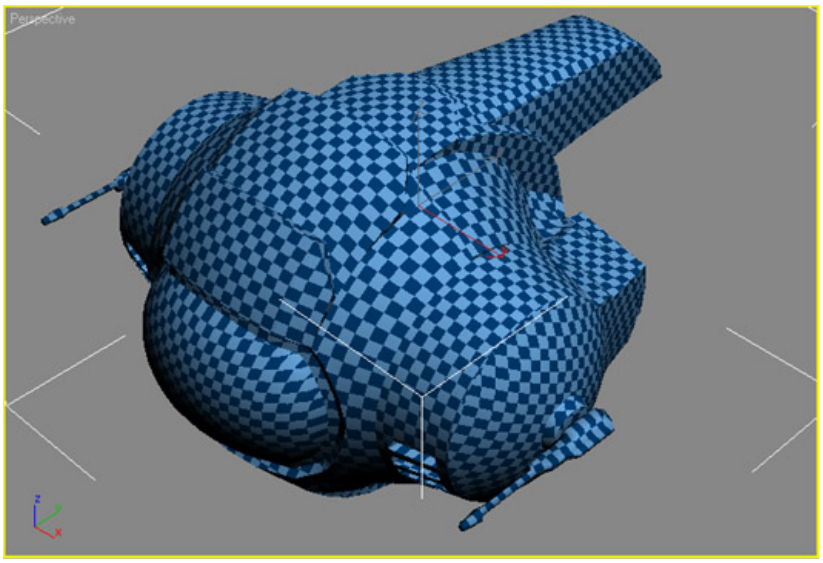

Abbildung 24: Modell gemappt

Das Mapping geschieht in 3ds Max. Das Normalmap-Baking kann ebenfalls in der 3D-Software erfolgen, oder aber nach einem anderen Algorithmus in der Software xNormal.

Die Texturierung erfolgt dann mit Photoshop bzw. den Photoshop-Plugins nDo2 und dDo. Diese von Quixel erstellten Unterprogramme können den Workflow erheblich vereinfachen und effizienter gestalten. Rost, Staub o.ä. Details müssen nicht mehr per Hand gestempelt werden, sondern werden prozedural anhand des Mappings generiert.

Durch dieses Kapitel ist somit die Pipeline für die Generierung der 3D-Assets klar definiert. Nun gilt es die Umgebung, in der das Assetkit eingebettet ist, näher zu betrachten.

3.3 Shark 3D Engine + Cinector

Wenn das Assetkit mit seinen Variationen, welches näher im folgenden Kapitel beschrieben wird, vollständig vorliegt, wird es in die Shark3D-Engine implementiert. Hierbei ist anzumerken, dass die Cinector-Modifikation, welche die Engine um zahlreiche Features erweitert, genutzt wird.

Bei Shark3D handelt es sich um eine Echtzeit-Engine, die auf einer Tree-Struktur fußt und damit einen nodebasierten Szenenaufbau ermöglicht.[36]Assets können mit Hilfe des Exporters, der von Spinor geschrieben wurde direkt aus 3ds Max in ein Format übertragen werden, was in die Engine implementiert werden kann. Shader- und Lightsetting erfolgt dann innerhalb der Engine.

Technische Begrenzungen innerhalb der Engine sind hierbei die Shader-Prefabs, die Anzahl der Texturen, die genutzt werden können und die Beleuchtung.

Es können zudem Animation und Kamerafahrten innerhalb der Software umgesetzt werden. Prozedurale Wettereffekte wie Regen oder Schnee sind ebenfalls möglich.

Die Modifikation Cinector wurde hierbei speziell für die Produktion von virtuellen Filmsets entwickelt und erweitert die Shark-3D-Engine insbesondere um Features im Animationsbereich.

Hierbei lassen sich virtuelle Kameras mit sämtlichen Settings realer Kameras einstellen und durch das interne Recording-Tool Szenen aufzeichnen. Wobei nicht das Bild sondern die Logik der Szene aufgenommen wird. Das Rapidbox-Prototyping gewährleistet zudem bestmögliche Arbeit für Regisseure zur Previsualisierung. Die modulare Bauweise, RapidBox, ist somit fester Bestandteil der Cinector-Philosophie und setzt den Anspruch der Nutzer an schneller und einfacher Bedienung um. Hieraus resultiert ebenfalls, wie bereits erwähnt, die hohe wirtschaftliche Relevanz.

Nachdem nun auch der Spielraum der Engine abgesteckt wurde und somit die Antwort auf die technische Begrenzung gegeben werden konnte, muss an dieser Stelle zu der generellen Frage nach der Konzeption und dem Arbeitsablauf einer Assetkit-Erstellung anhand des Praxisprojektes gefragt werden.

Wie soll das Thema „Haus im Wald“ umgesetzt werden? Wie wird das RapidSeasons-Konzept eingearbeitet?

[36] Vgl. Spinor GmbH, 2014

4 Workflow

4.1 Konzeption des Beispielprojektes

Innerhalb des Beispielprojekts wird, wie bereits erwähnt, das Setting „Haus im Wald“ bearbeitet, da es mit dem Grundbaustein – ein Gebäude – eine passende Testumgebung bildet. Dörfer, Städte oder eine Insel würden den Rahmen der Bachelorarbeit übersteigen. Anhand der Begrenzung auf ein einzelnes Szenario wird das Rapidbox-System zudem hinsichtlich der Elemente getestet:

Es muss mit wenigen Elementen eine große Wirkung erzielt werden. Zudem ermöglicht das Setting eine schnelle und einfache Möglichkeit das System auszuprobieren.

Im Rahmen der Konzeption gilt es hierbei folgende Punkte zu bestimmen:

Wie ist das Setting visuell konzipiert?

Das Setting „Haus im Wald“ vereint visuell Natur und Urban. Daher sind die vorherrschenden Materialen Erde, Holz, Metall, Glas und verschieden Steinarten. Je nach Asset finden sich zudem spezielle Oberflächen, wie Farbe oder Putz.

Es gilt dabei auch im Hinblick auf Kapitel 5 die genauen Rahmenbedinungen für das Environment zu klären, d.h.:

- Environment Setting (Nature+House)
- Location (Northern America/Countryside New York)
- Theme (Seasons> 1 Year)

Hierbei wird auf die Rapidbox Themenliste sowie die Enviroment-Setting- und Theme-Styles-Liste von Alex Galuzin zurückgegriffen. [37]

Generell gilt: je mehr Informationen über das Environment vorliegen, desto detailierter und auch realistischer kann es umgesetzt werden. Dies betont auch Tor Frick in seiner UDK Masterclass zu Modular Assetcreation[38].

[37] Vgl. Galuzin, 2011
[38] Vgl. Frick, 2013

Zusätzlich gilt zu klären, wie die Landschaft bzw. das Haus aufgebaut sein soll. Dafür liegt ein Grundriss vor:

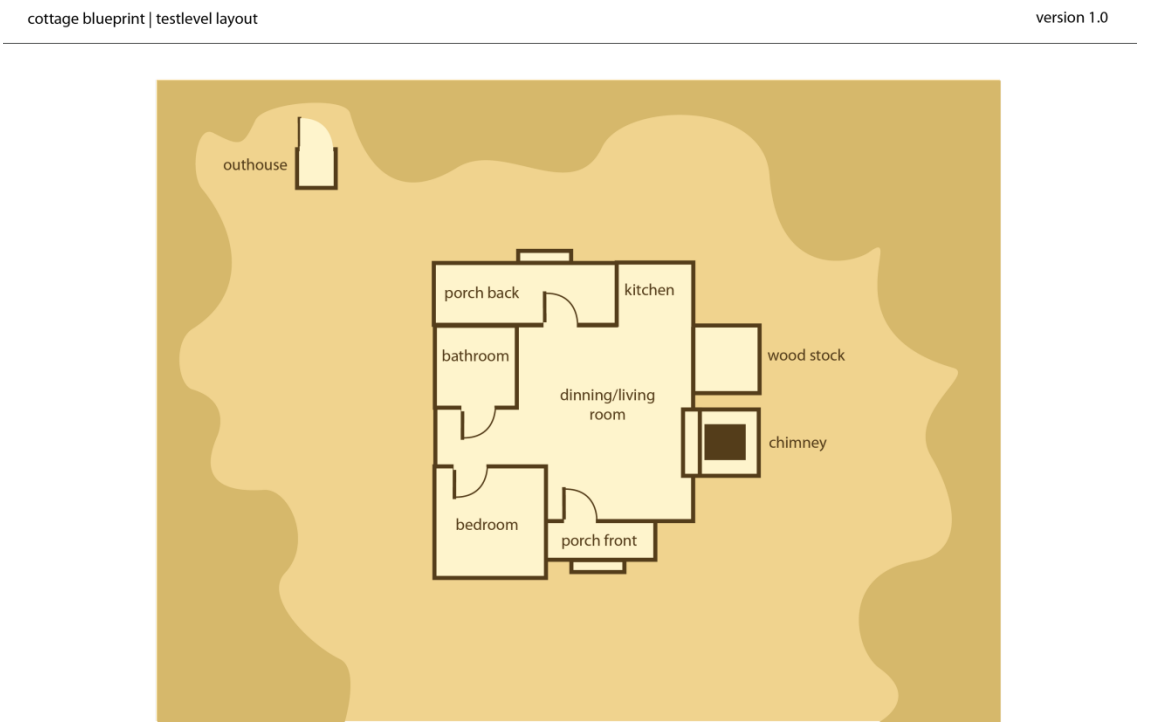

Abbildung 25: Cottage Blueprint

Damit ein Gefühl für Proportion und Raum entsteht, wird der Grundriss mit vereinfachter Geometrie in 3ds Max umgesetzt.

Visuell wird der Stil auf einen Mix[39] von Ranch, Cottage und Country festgelegt, wie es für Häuser in Nordamerika der Kolonialzeit üblich war, siehe Abbildung 25.

[39] Vgl. Hanley Wood, LLC., 2014

Abbildung 26: Stil-Beispiel

Grundlegende Kriterien für das Rapidboxkit müssen ebenfalls festgelegt werden, dazu zählen Grundgrößen, wie:

- Wanddicke
- Türmaße
- Fenstermaße

Das zu entwickelte Assetkit soll zudem Variationen ermöglichen, d.h. das Haus könnte auch mit einem anderen Grundriss aufgebaut werden. D.h. das Kit muss durch zusätzliche Props seinen repetitiven Charakter verlieren. Das Assetkit gliedert sich somit in das Coreassetkit für das Haus und die Landschaft (und als Teil dessen Subkits für Wege, Zäune und Ballustraten für die Vorbauten).

Es wird hierbei lediglich der Außenbereich des Hauses gebaut, da das Interior individuell zusammengestellt werden müsste.

Nachdem das Haus in seinem Grundriss, Architekturstil sowie notwendige Grundgrößen klar umrissen ist, muss zudem als Grundlage die Materialauswahl für das RapidSeason-System vorgenommen werden, d.h. welche Materialien verändern sich merklich innerhalb der Jahreszeiten. Ein Hauptaugenmerk muss dabei auf Holz gelegt werden, da dies das Material für die Wände darstellt und bei Feuchtigkeit quillt, sowie Metall, welches sich bei Wärme ausdehnt und bei Kälte zusammenzieht. Diese beiden Rohstoffe bilden somit einen ersten Ansatz für die Veränderung des Hauses in den Jahreszeiten.

Zudem wird das Haus nun mit der Basisserie nachgebaut und festgestellt inwieweit Teile für das Haus passend zu dem Rapidboxkonzept erstellt werden müssen. Auf dieser Basis findet auch die Aufteilung von Coreassetkit, Subkits und Props statt.

Für Letzteres wird hierbei eine Trennung zwischen Vegetation und Hausinventar(/Besitz) und RapidSeason vorgenommen.

Für jedes einzelne Asset wird nun bestimmt welche Materialien d.h. Shader+Texturen gebraucht werden sowie bei welchen Assets Texture Bakes notwendig sind, d.h. Erstellung eines High- sowie Lowpolys und die Generierung einer Normalmap vom Highpoly für das Lowpoly.

Dies muss insbesondere auch für die Rapidseason-Props vorgenommen werden. Im Kapitel 5.2 erfolgt hierbei die Auflistung der jahreszeitenbedingten Props.

Als nächster Schritt wird innerhalb der Konzeption, nachdem der visuelle Stil und Aufbau des Environments klar ist, die Umwelt in Augenschein genommen:

Gibt es Partikel-, Beleuchtungs- oder physikalische Effekte, für die Meshes, Texturen oder Skripte bereitgestellt werden müssen?

Da es sich bei einem virtuellen Filmset der Forschungsgruppe Gamecast auch um Animation dreht müssen zeitlich bedingte Effekte wie Tag und Nacht, Sonnenauf- und untergänge ebenfalls berücksichtigt werden.

Je nach Vorgabe d. Look&Feel kann dies z.B. auch eine entsättigte Farbenpalette für die Texturen und das Licht bedeuten, ein Beispiel dafür wäre z.B. die Gestaltung von „I am alive“

Abbildung 27: I am Alive Screenshot

Für das Thema Haus im Wald wurde hierfür eine Tonality[40] aufgestellt, d.h. Attribute für die vermittelte Stimmung definiert. Dies wird vorwiegend in der Werbung eingesetzt, um bestimmte Adjektive beim Rezipienten über ein Produkt zu kommunizieren.

Bei der Hütte im Wald ist dies vorallem:

- Idyllisch
- heimatlich
- naturbelassen
- lebendig
- friedlich

[40] Vgl. Marketing-Lexikon-Online, 2014

Dieser Grundton bzw. –stimmung kann als Ausgangspunkt für das Rapidseason-System bezüglich jeder Jahreszeit genutzt werden. Auf diese Art und Weise bleibt der ursprüngliche Charakter des Environments im Laufe der Texturierung und Bestimmung der Farbpalette erhalten und Fehltritte werden diesbezüglich verhindert.

Als letzter Schritt treten Partikel- bzw. physikalische Effekte in den Vordergrund.

Gibt es Effekte wie Wind, Feuer, Regen/Wellen oder Erde, die sich bewegen?

Im Fall vom Haus im Wald wären das die Wettereffekte, die innerhalb der Jahreszeiten eintreten. Zudem kommen während des Herbtes Glühwürmchen hinzu. Werden Nebel und Wolken ebenfalls innerhalb der Engine mit Partikeln gelöst, gehört dies ebenfalls dazu. Genauso wie sog. Godrays, die durch direktionales Licht durch z.B. Blätterwerk durchscheinen.

Zusammenfassend lässt sich sagen, dass die Konzeptionsphase insbesondere den Zweck erfüllt, die Assetgenerierung so genau wie möglich zu umreißen und bis ins Detail zu planen.

Außerdem entwickelt sich auf diese Art und Weise der visuelle Stil des Environments und auftretende Probleme bei z.B. dem Aufbau des Core-Assetkits kann entgegen gewirkt werden. Durch diese Schritte ist zudem eine genauere Zeitplanung möglich, als wenn sofort mit der Umsetzung begonnen wird.

Anhand dieses Kapitels wurde die Frage nach der Konzeption des themenspezifischen Settings beantwortet. Die Umsetzung bildet hierbei einen weiteren Kernpunkt, der nicht nur, wie in Kapitel 3 beschrieben in seinen Bedingungen und Begrenzungen klar sein muss, sondern auch in den einzelnen Produktionsstufen. Es stellen sich hierbei also insbesondere zwei Fragen:

1. Wie wird das Assetkit produziert?

2. Welche Zwischenschritte müssen innerhalb der einzelnen Tools beachtet werden?

4.2 Umsetzung

In diesem Projekt sind Bauweise – RapidBox – sowie der zuvor definierte visuelle Stil vorgegeben. Daher wurde mit Hilfe der vorgegebenen Basisserie und dem Grundriss, an dem sich das Haus orientiert, das Level geblockt. Dadurch wurde klar, welche Teile wo eingesetzt und welche Props integriert werden können. So entstand eine Tabelle

mit einer Einteilung in Corekit, Subkit, Props Vegetation und Props bzw. Vfx für Jahreszeiten. Das Hauptaugenmerk lag hierbei insbesondere bei Core- und Subkit, da hier die modulare Bauweise bedient werden musste. Es kristallisierte sich allerdings schnell innerhalb des Projektes heraus, dass nur durch Detaillierung der Assets bzw. Props Lebendigkeit innerhalb des Levels geschaffen werden konnte. Mit Hilfe einer Skizze aus der Vogelperspektive wurde zudem die Platzierung klar.

Die Arbeitsschritte innerhalb der Umsetzung teilten sich in die Erstellung der Basis- und Lowpoly-Geometrie in 3ds Max, die Zuweisung von UV-Koordinaten bzw. Mappen in 3ds Max, die Detaillierung der Objekte bzw. Erstellung des Highpolys in ZBrush, des Texture Bakings in xNormal bzw. 3ds Max und des Texturings in Photoshop bzw. mit Hilfe der Plugins nDo und dDo. Als Abschluss wurde die Datei für den Export in 3ds max vorbereitet und anschließend das Core- und Subkit sowie alle Props in Cinector implementiert.

Um einen näheren Einblick in die Erstellung eines Assetskits zu geben, soll hierbei ein Kurzabriss der verschiedenen Aufgaben und deren Reihenfolge innerhalb der Tools gegeben werden.

Bei der Modellierung der niedrig-polygonal aufgelösten Geometrie erfolgte ein immer wieder auftretendes Testen, ob die Teile modular funktionieren. So konnten bereits in dieser frühen Phase Schwachstellen gefunden und angepasst werden.

Zudem wurden alle Assets sowie Props gemappt und durch Zuweisung einer Kacheltextur überprüft. So ist es möglich Verzerrungen oder ungleiche Auflösungen von Teilen zu erkennen. Die Erstellung der Kollisionsmodelle erfolgte hierbei ebenfalls. Die Platzierung des Pivot-Punktes wurde zudem nach den zuvor erarbeiteten Ansätzen von Kapitel 2.1 vorgenommen. Eine Besonderheit ist hierbei, dass für die Vegetation aufgrund von Ressourcenersparnissen eine Erweiterung des Billboard-Modells genutzt wurde, um z.B. das Gras aus allen Winkeln visuell ansprechend zu gestalten.

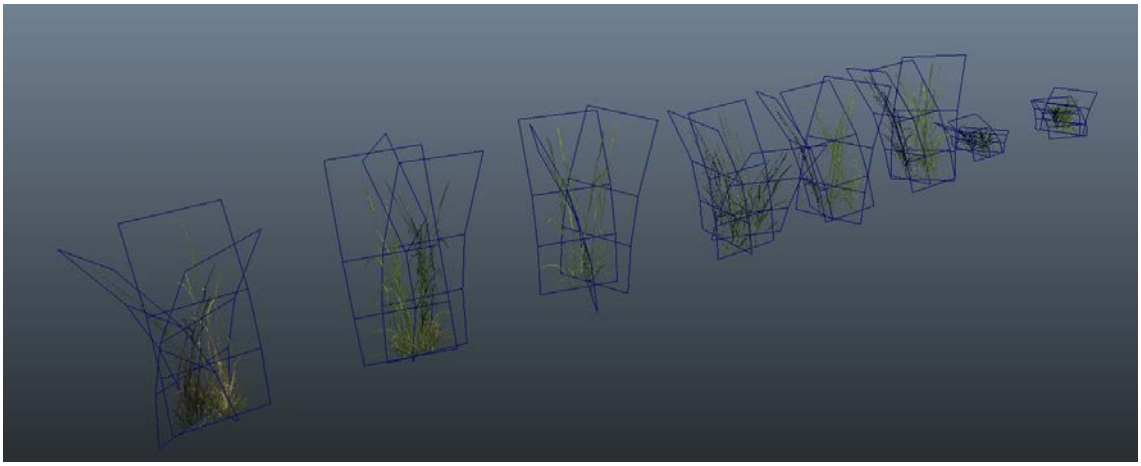

Abbildung 28: Erweitertes Billboard-Modell

War ein weiterer Schritt in ZBrush notwendig, wurde zudem ein Basismesh mit einer höheren und gleichmäßigen Segmentierung als die LowPoly-Variante erstellt. Diese wurde dann in ZBrush weiter bearbeitet und bei Bedarf höher segmentiert. Anschließend wurde durch einen programminternen Algorithmus die Polygonanzahl reduziert und das letztlich hochaufgelöste Modell in xNormal oder 3ds Max direkt für Texture Baking genutzt. Hierbei werden die Normaleninformationen des Highpolys auf das Lowpoly projiziert und eine Normalmap erstellt. Anhand dieser Map konnten zudem in nDo bzw. dDo weitere Maps generiert werden und in beliebigen Formaten abgespeichert werden.

Die so generierten Details wirken dem repetitiven Charakter entgegen. Teilweise entstanden auch Photonormalmaps, d.h. die Fototexturen wurden durch einen Algorithmus von nDo in Normalmaps umgewandelt und übernahmen somit den Zwischenschritt von ZBrush. Die Texturierung erfolgte in Photoshop mit den zuvor genannten Plugins nDo und dDo. Ein weiterer Aspekt auf den Rücksicht genommen werden musste ist, dass die Engine keine zweiseitigen Texturen besitzt und somit insbesondere bei den Props für Vegation planare Flächen kopiert und gespiegelt werden mussten, damit man innerhalb des Levels nicht durch sie hindurch sehen kann.

Nach erfolgreichem Modellieren, Mappen, Texturieren und Exportieren fand die Komposition des Levels in der Engine statt. Wobei auch die Kollisionsmodelle sowie die Platzierung des Pivot-Punktes überprüft werden konnten. Innerhalb der Engine geschah ebenfalls die Lichtgestaltung, welche auf Global Illumination basiert. Insbesondere das Echtzeitrendering der Engine erwies sich hierbei als großer Vorteil.

Zusammenfassend lässt sich somit sagen, dass verschiedene Auflösungen des 3D-Objektes in 3ds Max generiert wurden. Die Pivot-Point-Platzierung anschließend stattfand und auf die fehlenden zweiseitigen Texturen innerhalb der Shark geachtet werden

musste. Für weitere Detaillierungen wurde ZBrush hinzugezogen und somit ein Beitrag zur Aufhebung des repetativen Charakters geleistet werden konnte. Mit Hilfe des High- sowie Lowpolys erfolgte dann das Texture Baking mit xNormal. Die abschließende Texturkomposition erfolgte dann in dDo und Photoshop. Die Vorbereitung für den Export fand wiederum in 3ds Max statt.

Als Schlusspunkt für das gesamte vierte Kapitel und somit die Antwort auf die Fragen nach der Produktion bzw. Zwischenschritten innerhalb der Tools gilt die Implementierung in die Shark. Dieser Schritt lässt uns zu der Frage der Konzeption des RapidSeason-Systems zurückkehren.

5 System RapidSeasons

5.1 Witterungsauswirkungen und Umsetzung in das modulare System

Das System RapidSeasons erweitert das Corekit, die Subkits und die Props um die Jahreszeiten Frühling, Sommer, Herbst und Winter – es muss beachtet werden, dass dies auf der jeweiligen Klimazone basiert. Für jede Jahreszeit gilt es hierbei eine Tonalität festzulegen, die sich auf die bereits bestehende auswirkt (idyllisch, heimatlich, naturbelassen, lebendig und friedlich; Anm. d. Verf.).

- Frühling:
    - erwachend
    - fröhlich/blumig
    - wärmend
- Sommer:
    - sonnig
    - wolkenlos
    - strahlend
- Herbst
    - regnerisch
    - bewölkt/dunkel
    - bunt
- Winter
    - frostig/(kahl)
    - rein/weiß
    - kristallklar

Es gilt also bei jeder Jahreszeit den Grundtonus des Hauses sowie die saisonale Stimmung innerhalb der jeweiligen Assetkitvariation einzufangen.

Anhand dieser Tonalität ergeben sich demzufolge auch erste Tendenzen bezüglich der Gestaltung der Materialien. Als Beispiel dient hierbei Holz:

- Frühling: helleres und wärmeres/gesättigteres Holz als im Herbst und Winter. Die Farbintensität ist höher als im Herbst und Winter. Das Holz dehnt sich wieder mehr aus.

- Sommer: von der Farbintensität her gibt es einen Höhepunkt innerhalb des Sommers, da dort das meiste Licht auf das Material trifft. Der Farbkontrast ist hierbei ebenfalls am höchsten. Das Holz ist sehr trocken und spröde, dies muss man auch an der Holzstruktur erkennen. Es zieht sich zusammen, dies hat eine Auswirkung auf die Holzoberfläche.

- Herbst: das Holz vergraut optisch, durch den bewölkten Himmel und quillt zudem durch die Feuchtigkeit auf. Bei Regen wird eine Regentropfenschicht auf dem Material durch hohe Reflexion simuliert.

- Winter: das Holz ist mit Feuchtigkeit vollgesogen und Frost bildet sich auf der Oberfläche. Demzufolge ist die der Tonwert dunkler als im Herbst. Das Holz dehnt sich aus.

Zusätzlich zu dem Volumen, der Oberflächenbeschaffenheit und der Optik welche in erster Linie von den Lichtverhältnissen, der Feuchtigkeit und der Temperatur abhängig sind, kommen zudem Merkmale wie jahreszeitenspezifische Vegetation, Niederschlag, Wolkenbildung sowie Sonneneinstrahlung hinzu.

Generell gilt, dass das hierfür mit Hilfe von geografischen Standorten Sonnenstunden und Niederschlagsmengen der Jahreszeiten festgestellt werden kann. Es ist demzufolge auch möglich RapidSeason auf ein spezifisches Gebiet anzuwenden. Als Vergleichsgebiet für das Konzept des Kolonialhauses mit Stilmix von Cottage, Ranch und Country wurde hierfür die Finger Lakes Region gewählt. Dies ist klimatisch realistisch, da die Kolonialsiedlungen in Amerika an der Eastside zu finden sind und dieser Bereich 238 Meilen, ca. 383km, von New York entfernt ist.[41]

[41] Vgl. Finger Lakes Connection, 2014

Bezüglich der Vegetation[42] in der Four Finger Lakes Region liegt eine Laubwaldzone vor, die Ahorn, Buche, Linde und Ulme umfasst, wie auch den Schierling[43].

Es werden hierbei für die einzelnen Jahreszeiten zur Identifikation typische Pflanzen umgesetzt. Im Frühling sind das die Frühlingsblüher. Im Sommer der Löwenzahn. Der Herbst äußert sich vorwiegend mit Pilzwuchs und gefallenem Laub. Der Winter schließt jegliche Vegetation aus.

Die technische Umsetzung des RapidSeasons-Systems sieht zudem vor, dass durch die Namensgebung auf Basis der Basisserie die Jahreszeiten letztlich innerhalb der Engine per Drop-Down-Menü ausgewählt werden können und sich so die einzelnen Versionen für Frühling, Sommer, Herbst und Winder austauschen.

Zusammenfassend ließ sich in diesem Kapitel das System RapidSeasons umreißen und stellt nun die Frage nach der letztlichen Anwendung auf das Beispielgebäude mit dem generierten Assetkit.

5.2 Anwendung und Resultat des Konzeptes

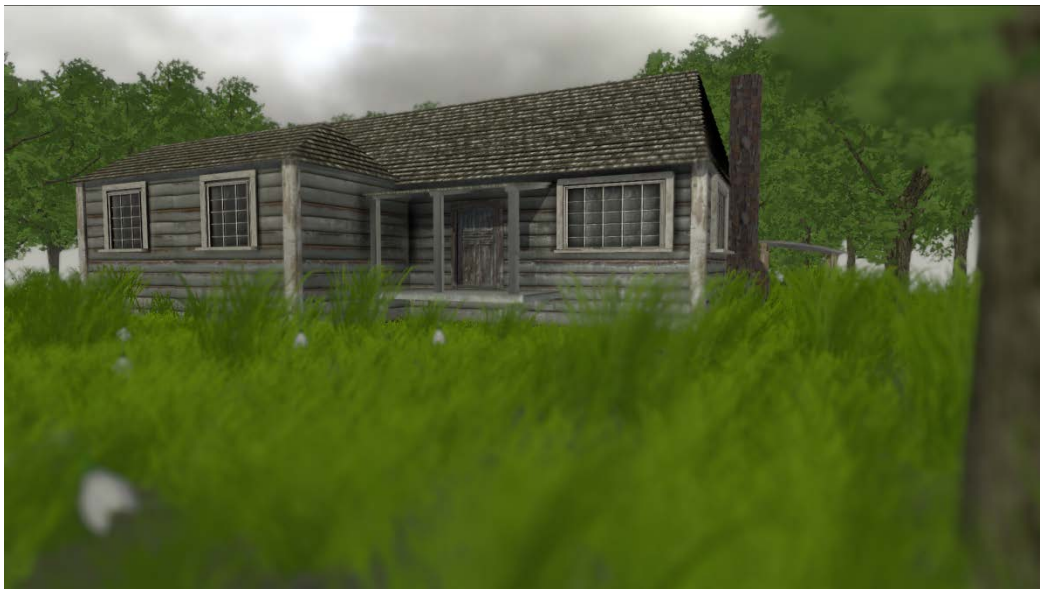

Abbildung 29: Frühling

[42] Vgl. Artdefects Media Verlag, 2014
[43] Vgl. Lewin, 1974

Durch das bereits festgelegte Setting und Veränderungen der Texturen für die einzelnen Jahreszeiten ergibt sich folgendes Ergebnis:

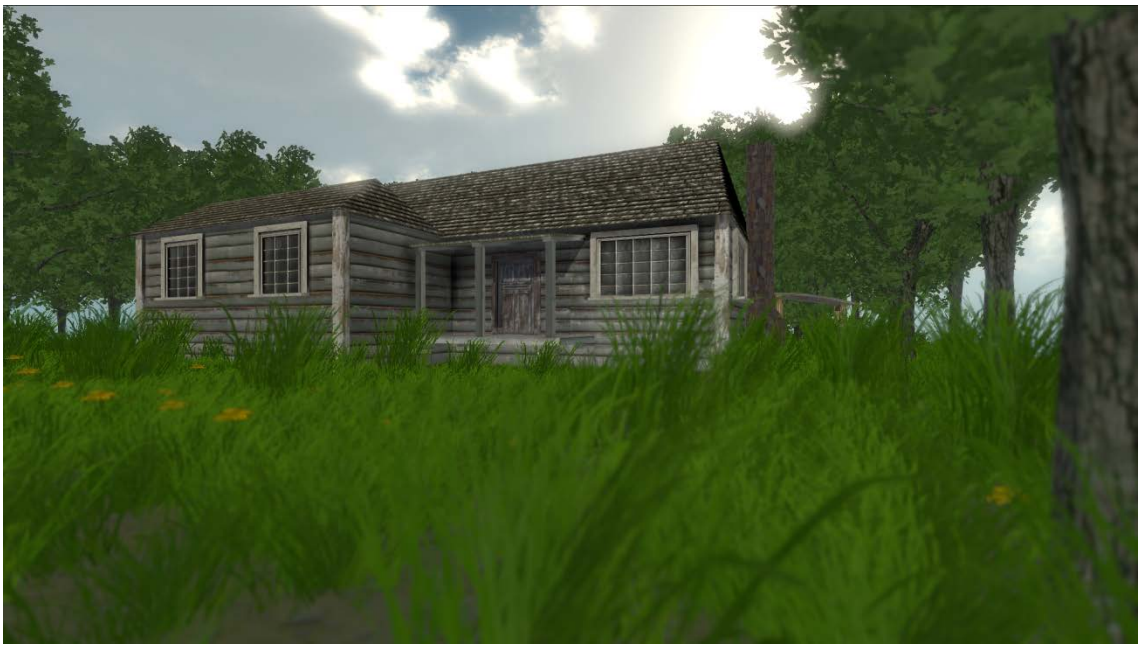

Abbildung 30: Sommer

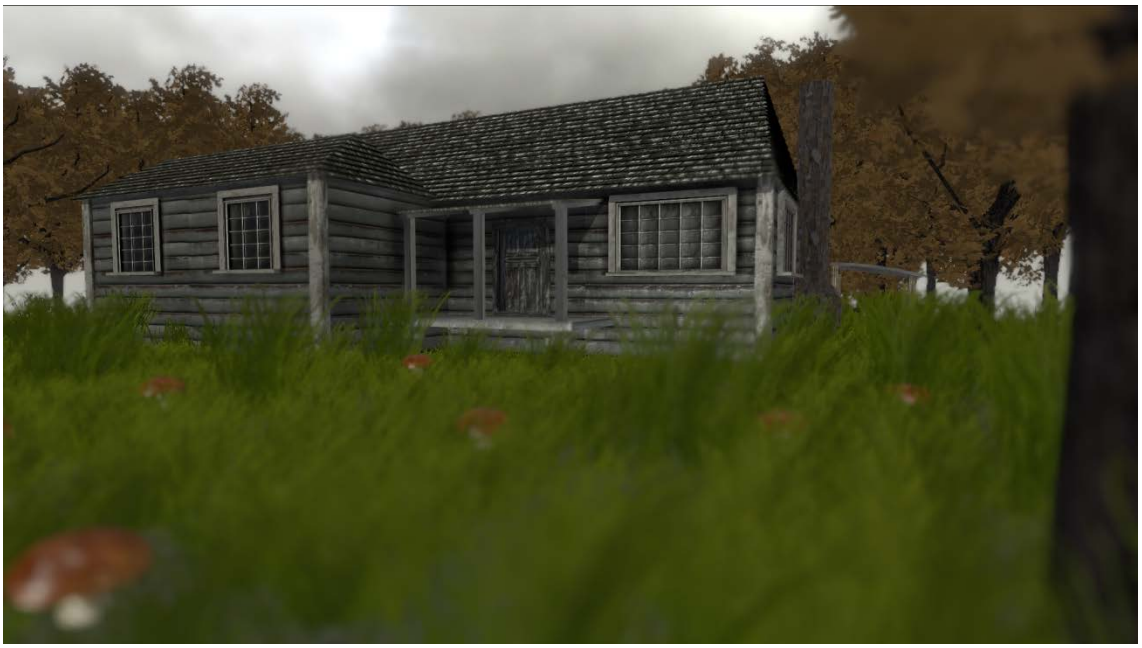

Abbildung 31: Herbst

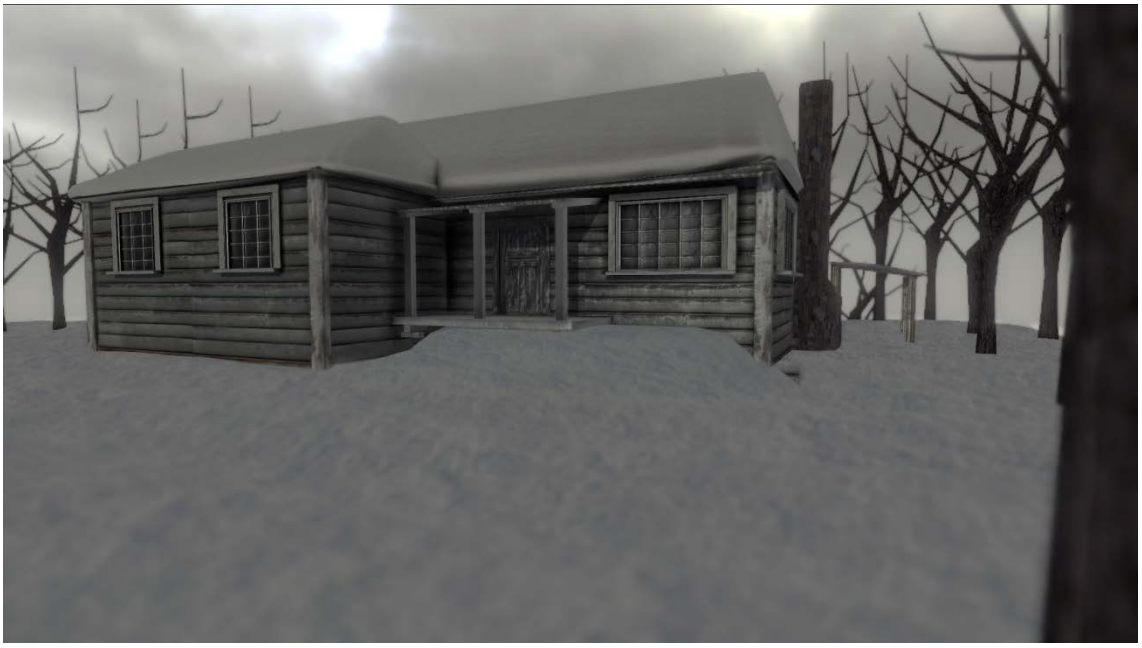

Abbildung 32: Winter

Insbesondere die Vegetation ist hierbei Ausdruck für die Jahreszeiten. Zusammen mit den Wettereffekten (der Shark) – Schnee, Wolken/Himmel, Lichtstimmung.

Durch die Auswahl der Jahreszeiten, die innerhalb des Engines möglich ist findet auch die zuvor aufgezeigte Zeitersparnis statt.

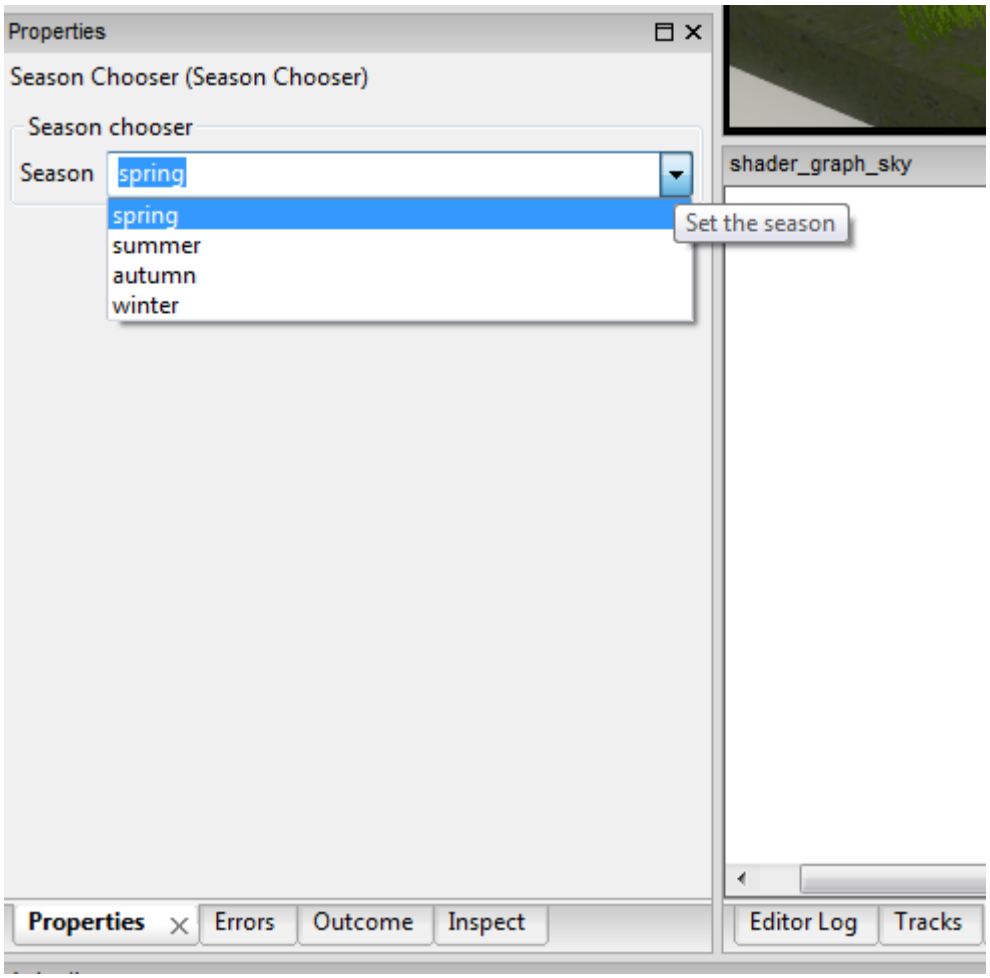

Abbildung 33: Season Chooser

Zusätzlich wurden für das Assetkit Props generiert.

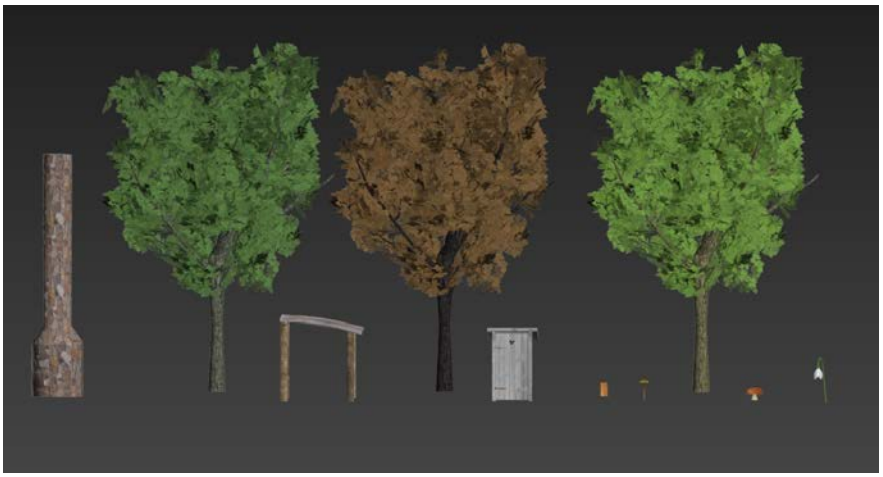

Abbildung 34: Props

Hierbei ist insbesondere auf die Lebendigkeit und Natürlichkeit eines Environments Rücksicht zu nehmen. Das hier vorliegende Assetkit hat lediglich eine begrenzte Anzahl von Teilen und besitzt lediglich ein Subkit. Würde nun z.B. der Wunsch bei einem Regisseur entstehen einen ganzen Wald mit diversen Hütten zu gestalten müssten mehr Teile produziert werden, um die Artenvielfalt darzustellen.

Ebenfalls wäre dann eine Überlegung bzgl. der Bodenelemente innerhalb von Natursettings ratsam, prozedurale Terrain-Generation in Betracht zu ziehen und mit den Assetkits zu kombinieren, um eine größere Vielfalt der Spielwelt und natürlichere Umgebung zu schaffen. Dies ließe sich auch, wie in Spielen mit einer Heightmap generieren und anschließend durch die Assetkits aufbrechen. Da Terrain-Generation sehr viele Polygone produzieren kann wäre es zudem ratsam ein Algorithmus, wie bei Halo Wars einzusetzen, der lediglich detaillierte Bereiche mit vielen Polygonen versieht und weniger definierte Bereiche geringer auflöst:

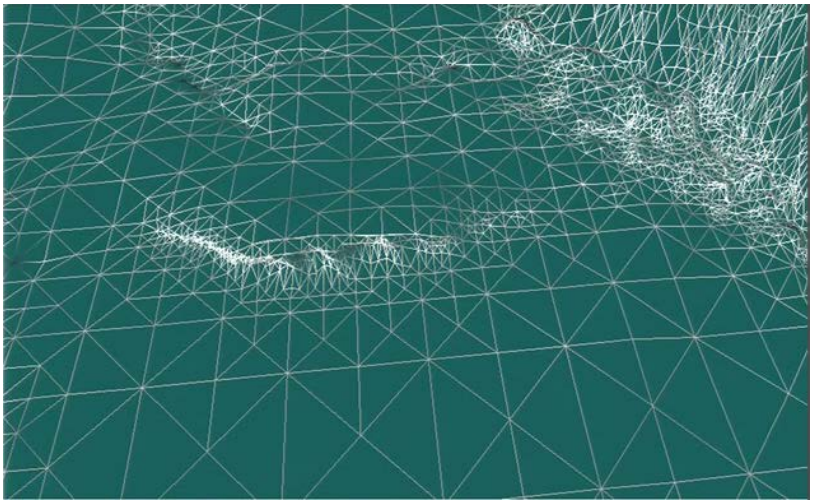

Abbildung 35: Halo Wars Terrain-Auflösung

Analysiert man Skyrims Environmentdesign auf der Webseite nexusmod, sind vorallem bei Gegenden mit Wald folgende Merkmale ersichtlich:

- Hügelige Landschaft
- Verschiedene Baum und Grassorten
    + Farbvariationen
    + unterschiedlich skaliert und gedreht
    + unterschiedliche Abstände zwischen der Vegetation

Abbildung 36: Skyrim Vegetation entlang des Weges

Abbildung 37: Skyrim Terrain innerhalb des Waldes

Eine Erweiterung der Shark-Engine wäre ebenfalls denkbar, sodass volumetrischer Nebel bzw. zweiseitige Texturen möglich werden. Damit könnte die Polygonanzahl bei Vegetation erheblich reduziert werden.

Die systematische Vorgehensweise, wie bei RapidSeasons nach Jahreszeiten zu kategorisieren könnte ebenfalls als Basis für weitere Module innerhalb von Cinector dienen. Diesbezüglich stellt sich hierbei nun die Frage wie RapidSeasons weiterverwendet werden kann, um nicht nur die Szenengestaltung eines virtuellen Filmsets zu erweitern – sondern mehrere virtuelle Filmsets innerhalb einer virtuellen Welt.

Im nächsten Kapitel wird dieser Frage nachgegangen.

6 Einsatz in der Zukunft

6.1 Asset-Zustände in Cinector

Durch die Anwendung von Rapid Season konnte gezeigt werden, dass die Vergabe von Assetzuständen eine Möglichkeit zur Variation von Assetkits bietet. Doch wie kann dieses Wissen in der Zukunft genutzt werden? Welche Zustände könnten noch implementiert werden?

Eine Option wäre das System RapidDecay, was auf verschiedenen Verfallsstufen der Objekte beruht und somit RapidSeasons ergänzen könnte und die Alterung von z.B. Gebäuden oder deren Abriss darstellen könnte.

Ein Beispiel wäre dafür eine Mauer mit folgenden Verfallskategorien:

1. Mauer, vollständig, keine Schäden
2. Mauer, erste Risse, Flecken, leichte Abplatzungen (1-3cm)
3. Mauer, tiefere/längere Risse, Flecken, abgeplatzter Putz (5-10cm)
4. Mauer, längere Risse, Flecken, abgeplatzter Putz mit offenem sichtbaren Mauerwerk
5. Mauer, große abgeplatzte Bereiche und offenes Mauerwerk, erste Steine liegen herum
6. Mauer, kleinere Löcher, Steine liegen zerbrochen rum
7. Mauer, große Löcher: an Rändern verwittert/dunkel, wenig Putz, Risse
8. Mauer, Ruine: nur noch 1/4-1/2 Mauerhöhe vorhanden, verwittert>komplett verfärbt, nicht mehr tragend

Während sich RapidDecay und RapidSeason auf die Gestaltung der Objekte beziehen wäre es zudem denkbar ein System zu integrieren, welche die Funktion der Assets näher beschreibt. Mit der Funktionalität ist hierbei insbesondere die Auslösung bzw. das Abspielen von Animationen und damit verbundene Sounds gemeint.

Da es innerhalb des Assetkits verschiedene Arten von Objekten gibt, bietet es sich an diese Objekte zu klassifizieren. Hierbei wird eine Unterteilung in Active und Passive

Objects vorgenommen. Diese werden als Active Objects in Interaktions-, Aktions- und Darstellungsobjekte eingeteilt. Eine Übersicht der Klassifizierung folgt:

- RapidFunction „Funktion – Animation“
    - Active Objects
        - Interaktionsobjekte>Animationsauslösend bei Charakteren und/oder Objekten
        - Aktionsobjekte> Nichtanimationsauslösend bei Charakteren, können es aber bei Objekten sein
        - Darstellungsobjekte> zeigen Umwelteinflüsse v, Nicht-Charakter-Einflüssen wie Wind, Wasser, Sonne, Erde, Feuer (Animations-loops für Lagerfeuer o.ä.)

Die Passive Objects können hierbei physikalisch eine Rolle spielen, d.h. durch Kollision Charaktere blockieren oder ohne Auswirkungen auf Charaktere oder Objekte sein, dann zählen sie in die Kategorie Physical-Non-Action-Objects.

    - Passive Objects
        - Physical--Action-Objects> Wände, die lediglich durch die Kollision die Charaktere „blockieren“
            - Physical-Non-Action-Objects> Objekte, die sich weder bei den Umwelteinflüssen ändern, noch bei Character-Interaktion

Es wäre hierbei ebenfalls zu überlegen, ob für die Assetproduktion wie in dem CryEngine-Exporter ein Algorithmus eingebunden wird, der die Kollisionsboxen automatisch erstellt und sich hierbei an ausgewählter Geometrie orienteirt. Ebenfalls denkbar wäre es in Zukunft Submats in die Shark einzuführen, damit die Engine weiß, welches Material z.B. mit einer Patronenkugel in Berührung kommt und somit Kratzer oder Einschusslöcher in diesem Bereich mit der zum Material passenden Textur zeigt.

Wie bereits voran gegangen kann die Denkweise von RapidSeasons auch in Systemen wie RapidDecay oder RapidFunction verwendet werden, um Assets nicht nur visuell zu kategorisieren sondern auch in ihrer Funktion näher zu beschreiben.

Doch gibt es eine Möglichkeit dies mit einem Prinzip für Zufallsgenerierung zu kombinieren?

6.2 Asset-Random-Generation

Um nicht nur zuvor erwähnte Systeme zu integrieren, sondern die Erstellung virtueller Filmsets in Cinector weiter zu automatisieren, wäre eine zufällige Generierung von Chunks möglich, d.h. Assetkits werden genutzt um Units zu erstellen und diese in der nächst größeren Einheit, den Chunks, zusammenfassen. Bei einer Stadt wäre so eine Unit ein ganzes Gebäude und ein Chunk ein Straßenblock. Prozedurale Generierung von Welten bildet im 3D-Design bereits in Titeln wie Minecraft[44], Cubeworld[45] oder das kommende Spiel No Man's Sky[46] einen Schwerpunkt. Dies könnte ebenfalls auf die Erstellung von virtuellen Filmsets angewandt werden.

Die Chunks würden hierbei themenspezifisch generiert werden und so z.B. auch den Übergang einer Waldhöhle in eine Eishöhle möglich machen. Den Ansatz des Kitbashings von Bethesda könnte ebenfalls eingearbeitet werden und somit Crossover-Gebiete zwischen themenspezifischen Chunks zu erstellen, wodurch anhand der zuvor definierten Asseteigenschaften bzw. –zuständen eine Anpassung an das andere Assetkit möglich wird. Die Waldhöhle hätte in dem Crossoverbereich so z.B. schon Eiszapfen statt Stalagniten und eine Eisdecke, zusätzlich wäre ihre Jahreszeit auf Winter gestellt.

Durch dieses System könnte in kürzerer Zeit als durch Assetkits eine Mehrzahl an virtuellen Filmsets entstehen und somit für den Regisseur nicht nur Zeitersparnis sondern auch Inspirationsquelle bieten.

Jedoch wäre dies ebenfalls an bestimmte Regeln gebunden, damit keine Fehler innerhalb der Generierung passieren. Als Beispielansatz ist in der folgenden Grafik auf Basis der 8x8m Bodenfläche genutzt worden.

Demnach ist ein Chunk in diesem Fall quadratisch aufgebaut und könnte ebenfalls verschiedene Größen haben.

8 Einheiten = größte Randlänge →48x48m Fläche→ 24 Einheiten Wall 4 Einheiten Corner→Rahmen 42 Innen-Teile

Gesamt: 72 Teile

---

[44] Vgl. Notch Developement AB, 2014
[45] Vgl. Funck, 2014
[46] Vgl. Amini, 2014

6 Einheiten = größte Randlänge →36x36m Fläche→ 16 Einheiten Wall 4 Einheiten Corner→Rahmen 16 Innen-Teile

Gesamt: 36 Teile

4 Einheiten = größte Randlänge →16x16m Fläche→8 Einheiten Wall 4 Einheiten Corner→Rahmen 4 Innen-Teile

Gesamt: 16 Teile

Die Aufteilung der Assets erfolgt hierbei nach der Bodenart. Im Randbereich des Chunks müssen zudem Übergangsteile vorhanden sein, die den Übergang zu anderen Kits bestimmen.

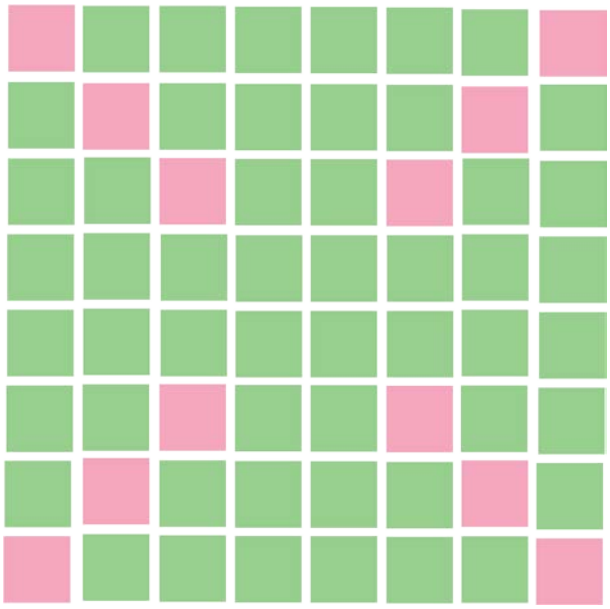

Abbildung 38: Chunkanordnung

Der Dateiname legt folgendes fest: Theme, Setting und Jahreszeit.

Dies ist ebenfalls mit dem fiktiven System RapidDecay erweiterbar. RapidDecay und RapidSeasons und würden hierbei direkt bei dem Asset auswählbar sein, während RapidFunction sich im Propertiesmenü des Assets befinden würde.

Eine Einteilung der Chunks bzgl. der Entscheidung ob sie Outerior und Interior abbilden ist ebenfalls ratsam, damit lediglich zusammenpassende Chunks angrenzen.

Abschließend ergibt sich somit in diesem Kapitel ein Ansatz, um mögliche Chunks so anzuordnen, dass sie prozedural generiert werden können.

7 Zusammenfassung und Resümee

7.1 Zusammenfassung

Ziel der Bachelorarbeit war es die visuelle Eintönigkeit eines modularen Assetkits durch die Konzeption und Umsetzung eines Jahreszeitensystems aufzubrechen.

Hierbei entwickelte sich ein Fragenkatalog der innerhalb dieser Arbeit beantwortet werden konnte:

1. Inwiefern hat sich Modular Asset Creation im aktuellen Kontext weiterentwickelt?

Waren für die Vertreter Mader, Perry, Kinney und Wanlass noch reine Funktionalität der Schwerpunkt bei Modular Asset Creation, so erweiterte sich innerhalb des Bethesda-Workflows zu Testing bestimmter Szenarien und der Integration von neuen Methoden wie Kitbashing oder Bauweisen wie das „Pivot and Flange“- und De-Twist-System.

2. Wie muss ein Konzept zur Erstellung eines Assetkits aussehen?

Das Konzept zur Erstellung eines Assetskits findet sich in Abbildung 15 wieder. An erster Stelle steht hierbei die Konzeption mit der Themenwahl, dann folgt der Test des modularen Systems. Anschließend erfolgt die Erstellung des Kits und zuletzt seine Korrektur bzw. Polishing.

3. Welche Rahmenbedingungen bzw. technische Grenzen bestehen hierbei bei der Umsetzung?

Die technischen Rahmenbedingungen sind individuell von der gewählten Engine abhängig und dementsprechend müssen Assets ebenfalls aufbereitet werden. Selbiges gilt für die Texturimplementierung.

4. Wie gestaltete sich die Konzeption von RapidSeasons?

In erster Linie wurde die Beispielszene an sich durch Environment Setting, Location und Theme definiert. Regeln wie Fenstermaße, Türmaße und Wanddicke müssen ebenfalls festgelegt werden. Hierbei wurde die Tonality für das Corekit an sich und für die Jahreszeiten festgelegt. Daraus erfolgte die Ableitung der genutzten Farbschemen sowie Lichtstimmung.

5. Wie und mit welchen Tools wurde RapidSeasons umgesetzt?

Es wurde mit 3ds Max die Grundgeometrie sowie weitere Auflösungen für die Erstellung des Highpolys in ZBrush geschaffen. Das Texturebaking erfolgte mit xNormal. Abschließend fand die Texturierung in Photoshop durch die Plugins nDo und dDo statt. Als letzter Schritt wurden die Assets in 3ds Max sharkready exportiert und letztlich in die Engine eingebunden. Dort fand noch die Lichtgestaltung statt.

6. Wie könnte ein Einsatz von RapidSeasons in der Zukunft aussehen?

Der Ansatz von RapidSeasons könnte mit weiteren Systemen wie RapidDecay und RapidFunction kombiniert werden.

7. Gibt es eine Möglichkeit dieses Prinzip mit einer Zufallsgenerierung zu kombinieren?

Werden aus dem modularen Assetkit Units bzw. Chunks erstellt, und die erläuterten Regeln eingehalten könnte eine Prozedurale Generierung ala Minecraft erfolgen. Somit wäre es auch möglich thematisch verschiedene Assetkits miteinander zu kombinieren.

Innerhalb dieser Arbeit konnte also, in Hinblick auf bereits praktizierte Workflows ein Konzept zur Erstellung eines Assetkits erarbeitet werden. Zusätzlich wurde RapidSeason ans Jahreszeitensystem konzipiert und das Konzept für zwei weitere Systeme – RapidDecay und RapidFunction, erarbeitet, welches ebenfalls mit RapidSeasons kombiniert werden kann. Im Hinblick auf die Zufallsgenerierung einer virtuellen Umgebung erfolgte ebenfalls eine Aufschlüsselung eines Chunk-Systems, was prozedurale Zufallgenerierung möglichen machen könnte.

7.2 Resümee

Nach Abschluss der Konzeption und Implementierung des modularen Systems Rapidbox mit der Erweiterung RapidSeasons stellt sich vorallem eine Frage:

Wie kann überprüft werden, ob durch RapidSeasons die visuelle Eintönigkeit eines modularen Assetkits tatsächlich aufgebrochen werden kann?

Dies ist ein Ansatz für eine Umfrage in den verschiedene Szenen in den verschiedenen Jahreszeiten dargestellt werden und mit einem Fragenkatalog und einer repräsentativen Teilnehmerzahl durchgeführt werden. Zudem müssten Kriterien für visuelle Qualität und Lebendigkeit definiert werden. Dies könnte in einer Masterarbeit näher untersucht werden.

Ein weiterer Ansatzpunkt ist der Ablauf des Workflows. So wäre es möglich in einem größeren Zeitrahmen die Produktionsphase zeitlich zu optimieren und verschiedene Tools wie Blender, Modo, ZBrush, 3ds Max, Cinema4D und Maya zu testen.

Im momentanen Status des Projektes, siehe Kapitel 5.2, können zudem noch weitere Assets bzw. Props, wie zuvor im Konzept beschrieben angefertigt werden. Die Integration der Wettereffekte, sowie Eiszapfen und fallendes Laub – möglicherweise prozedural gelöst – könnte hierbei der nächste Schritt sein.

Zudem könnten weitere Varianten des Corekits angefertigt werden, um die Anzahl der möglichen Variation mit der Rapidbox-Bauweise zu testen und die Basisserie auf diese Art und Weise zu erweitern.

Literaturverzeichnis

Aisslinger, W. (kein Datum). *loftcube*. Abgerufen am 04 2014 von http://www.loftcube.net/

Amini, T. (18. 06 2014). *kotaku*. Abgerufen am 22. 06 2014 von http://kotaku.com/how-a-seemingly-impossible-game-is-possible-1592820595

Artdefects Media Verlag. (2014). *usa-greencard*. Abgerufen am 26. 05 2014 von http://www.usa-greencard.eu/klima_vegetation_und_tierwelt_in_den_usa.html

Bohn, D. (15. 04 2014). *theverge*. Abgerufen am 22. 04 2014 von http://www.theverge.com/2014/4/15/5615880/building-blocks-how-project-ara-is-reinventing-the-smartphone

Burgess, J. (19. 04 2013). *blog.joelburgess*. Abgerufen am 24. 04 2014 von http://blog.joelburgess.com/2013/04/skyrims-modular-level-design-gdc-2013.html

Büttner, M. (2011). *x17*. Abgerufen am 22. 04 2014 von http://www.x17.de/

Cinector GbR. (2013). *Rapid-Box Aufbauprinzip.* Mittweida.

Epic Games, Inc. (2001-2012). *udn.epicgames*. Abgerufen am 23. 04 2014 von https://udn.epicgames.com/Three/ModularEnvironmentCreation.html

Finger Lakes Visitors Connection. (2014). *visitfingerlakes*. Abgerufen am 27. 05 2014 von http://visitfingerlakes.com/about-the-region/weather/

Frick, T. (2013). *eat3d*. Abgerufen am 26. 05 2014 von http://eat3d.com/udk_modular

Funck, W. v. (2014). *picroma*. Abgerufen am 12. 06 2014 von https://picroma.com/cubeworld

Galuzin, A. (2011). *Preproduction Blueprint.* World of Level Design.

Greenough, H. (1947). *Form and Function: Remarks on Art.* Berkeley: University of California Press.

Hanley Wood, LLC. (2014). *builderhouseplans*. Abgerufen am 26. 05 2014 von http://www.builderhouseplans.com/a-wonderful-welcome-for-guests/pid/114111271

Jelsoft Enterprises Ltd. (2000-2014). *polycount*. Abgerufen am 24. 04 2014 von http://wiki.polycount.com/ModularMountAndBlade?action=AttachFile&do=view&target=Modular_MountBladeMod_03.jpg

Joel Burgess, N. P. (26. 05 2013). *slideshare*. Abgerufen am 14. 05 2014 von http://de.slideshare.net/JoelBurgess/gdc2013-kit-buildingfinal

John K. Gershenson, G. j. (1999). Modular Product Design: A Life-cycle View. *Journal of Integrated Design and Process Science*, 1-3.

Johnson, A. (07. 05 2014). *docs.cryengine*. Abgerufen am 24. 05 2014 von http://docs.cryengine.com/display/SDKDOC3/Art+Asset+File+Types

Jones, S. (2011). *Investigation into modular design within computer games.* Staffordshire: Staffordshire University.

Keller, E. (2011). *Introduction ZBrush 4.* Indianapolis: Wiley Publishing, Inc.

Kinney, J. (1. 12 2012). *digitaltutors*. Abgerufen am 25. 04 2014 von http://www.digitaltutors.com/tutorial/857-Asset-Workflows-for-Modular-Level-Design

LEGO Group. (2014). *lego*. Abgerufen am 23. 05 2014 von http://www.lego.com/de-de/products

Lewin, D. C. (1974). *The Vegetation of the Ravines of the Southern Finger Lakes.* Iowa: Iowa State University.

Mader, P. (02. 12 2005). *gamasutra*. Abgerufen am 02. 05 2014 von http://www.gamasutra.com/view/feature/2475/creating_modular_game_art_for_fast_.php

Marketing-Lexikon-Online. (2014). Abgerufen am 26. 05 2014 von http://www.marketing-lexikon-online.de/index.php?option=com_content&view=article&id=311:tonality&catid=1:lexikon

Notch Developement AB. (2014). *minecraft*. Abgerufen am 27. 06 2014 von https://minecraft.net/

Perry, L. (2002). Modular Level and Component Design. *gd mag*, 1-6.

Scott, R. (2014). *nexusmod*. Abgerufen am 27. 06 2014 von http://www.nexusmods.com/skyrim/

Seric, A. (03. 03 2014). *ashleyseric*. Abgerufen am 22. 05 2014 von http://ashleyseric.com/unity-vs-udk/

Shotgun Software Inc. (2013). *shotgunsoftware*. Abgerufen am 24. 05 2014 von https://www.shotgunsoftware.com/features/

Smith, R. E. (2010). *Prefab Architecture.* New Jersey: John Wiley & Sons.

Spinor GmbH. (2014). *spinor*. Abgerufen am 26. 05 2014 von http://www.spinor.com/features.html

Thorn, A. (2011). *UDK Game Development.* Kentucky: Course Technology PTR.

Ullrich, C. (25. 04 2006). *focus*. Abgerufen am 07 2014 von http://www.focus.de/immobilien/bauen/fertighaus/fertighaus_aid_18063.html

Wanlass, T. (2010-2014). *3dmotive*. Abgerufen am 12. 05 2014 von https://www.3dmotive.com/f101001

Wedekind, K. (22. 04 2014). *n-tv*. Abgerufen am 23. 04 2014 von http://www.n-tv.de/technik/Baukasten-Handy-kommt-im-Januar-article12698151.html

wikipedia. (16. 05 2014). Abgerufen am 24. 05 2014 von http://de.wikipedia.org/wiki/Apache_Subversion

Eigenständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe. Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt.

Ort, Datum Vorname Nachname